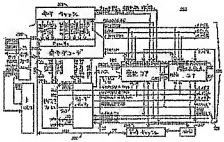
SUPERSCALAR MICROPROCESSOR

Publication number: JP7182160 (A)
Publication date: 1995-07-21
Inventor(s): DEIBITSUDO BII UITSUTO; UIRIAMU EMU JIYONSON +
Applicant(s): ADVANCED MICRO DEVICES INC +
Classification:
- international: *G06F9/30; G06F9/302; G06F9/32; G06F9/38;* G06F9/30; G06F9/302; G06F9/32; G06F9/38; (IPC1-7): G06F9/38; G06F9/38
- European: G06F9/302; G06F9/30F; G06F9/30T2; G06F9/32C; G06F9/38E; G06F9/38E2; G06F9/38F; G06F9/38F2B; G06F9/38S6; G06F9/38T
Application number: JP19940263317 19941027
Priority number(s): US19930146382 19931029

Also published as:
JP3670039 (B2)
US5651125 (A)
US5574928 (A)
EP0651321 (A1)
EP0651321 (B1)
more >>

Abstract of JP 7182160 (A)

PURPOSE: To process instructions in parallel by providing a common register file for receiving an already used instruction result from a common reorder buffer used by both an integer functioning unit and a floating point functioning unit.
CONSTITUTION: Relating to the architecture of a microprocessor 200, an integer functioning unit 215 and e floating point functioning unit 225 include plural waiting stations 220 and 230, and they are connected with a common data processing bus 535. Then, the integer functioning unit 215 and the floating point functioning unit 225 commonly use a reorder buffer 240. An already used instruction result is received by a common register file 235 from the reorder buffer 240. Thus, the input of orderly instructions and the execution of the disorderly instructions are processed in parallel.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平7－182160

(43)公開日　平成7年(1995)７月21日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　9/38	３１０　Ｆ			
	３５０　Ａ			

審査請求　未請求　請求項の数27　ＯＬ　（全 46 頁）

(21)出願番号　特願平6－263317

(22)出願日　平成6年(1994)10月27日

(31)優先権主張番号　１４６３８２

(32)優先日　1993年10月29日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591016172
アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド
ADVANCED MICRO DEVICES INCORPORATED
アメリカ合衆国、94088－3453 カリフォルニア州、サニィベイル、ピィ・オゥ・ボックス・3453、ワン・エイ・エム・ディ・プレイス（番地なし）

(74)代理人　弁理士　深見　久郎　（外３名）

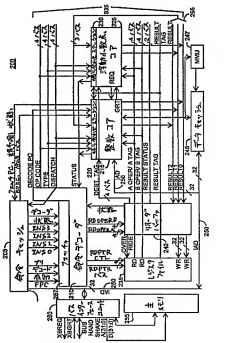
最終頁に続く

(54)【発明の名称】　スーパースカラマイクロプロセッサ

(57)【要約】

【目的】　高性能のスーパースカラマイクロプロセッサを提供する。

【構成】　上記マイクロプロセッサ２００は、高性能主データ処理バスを共有する整数機能ユニットと浮動小数点機能ユニットとを含む。整数ユニットと浮動小数点ユニットは、共通リオーダバッファ、レジスタファイル、分岐予測ユニットおよびロード／ストアユニットも共有し、これらはすべて同じ主データ処理バス上にある。命令およびデータキャッシュが主メモリに、この間の通信を扱う内部データアドレスバスを介して結合される。命令デコーダが命令キャッシュに結合され、１マイクロプロセッササイクルにつき複数の命令をデコードすることができる。命令は推論順にデコーダから発行され、投入および完了は順序通りでない。命令はリオーダバッファからレジスタファイルに順序通りに格納される。機能ユニットは複数のデータ幅を示すオペランドを収容する。



【特許請求の範囲】
【請求項１】　スーパースカラマイクロプロセッサであって、
同じマイクロプロセッササイクル中に複数の命令をデコードするための複数命令デコーダを含み、前記デコーダは同じマイクロプロセッササイクル内に整数および浮動小数点命令の両方をデコードし、さらに前記デコーダに結合されるデータ処理バスと、
前記データ処理バスに結合される整数機能ユニットと、前記データ処理バスに結合される浮動小数点機能ユニットと、
前記データ処理バスに結合されて、前記整数機能ユニットおよび前記浮動小数点機能ユニットの両方によって用いられる共通リオーダバッファと、
前記リオーダバッファに結合されて、前記リオーダバッファから用済とされた命令結果を受入れる共通レジスタファイルとを含む、スーパースカラマイクロプロセッサ。
【請求項２】　前記整数機能ユニットが少なくとも１つの待合わせステーションを含む、請求項１に記載のマイクロプロセッサ。
【請求項３】　前記整数機能ユニットが２つの待合わせステーションを含む、請求項１に記載のマイクロプロセッサ。
【請求項４】　前記浮動小数点機能ユニットが少なくとも１つの待合わせステーションを含む、請求項１に記載のマイクロプロセッサ。
【請求項５】　前記浮動小数点機能ユニットが２つの待合わせステーションを含む、請求項１に記載のマイクロプロセッサ。
【請求項６】　前記データ処理バスが、
複数のｏｐコードバスと、
複数のオペランドバスと、
複数の命令タイプバスと、
複数の結果バスと、
複数の結果タグバスとを含む、請求項１に記載のマイクロプロセッサ。
【請求項７】　前記オペランドバスがオペランドタグバスを含む、請求項６に記載のマイクロプロセッサ。
【請求項８】　前記データ処理バスが予め定められたデータ幅を示し、前記リオーダバッファが、前記データ処理バス幅に等しい幅を示すエントリと、前記データ処理バスのデータ幅の倍数に等しい幅を示すエントリとをストアするメモリ手段を含む、請求項１に記載のマイクロプロセッサ。
【請求項９】　前記デコーダが、プログラム順に整数および浮動小数点命令の両方を発行するための発行手段をさらに含む、請求項１に記載のマイクロプロセッサ。
【請求項１０】　前記データ通信バスに結合されて、前記整数機能ユニットと前記浮動小数点機能ユニットによって共有される分岐予測機能ユニットをさらに含む、請求項１に記載のマイクロプロセッサ。
【請求項１１】　前記浮動小数点機能ユニットが、複数のサイズを示すオペランドを処理する、請求項１に記載のマイクロプロセッサ。
【請求項１２】　前記浮動小数点機能ユニットが、単精度／倍精度浮動小数点機能ユニットを含む、請求項１に記載のマイクロプロセッサ。
【請求項１３】　前記複数命令デコーダが、１マイクロプロセッササイクルにつき４つの命令をデコードすることができる、請求項１に記載のマイクロプロセッサ。
【請求項１４】　前記マイクロプロセッサを、命令およびデータがストアされる外部メモリにインタフェースさせるためのバスインタフェースユニットと、前記バスインタフェースユニットに結合される内部アドレスデータ通信バスと、
前記データ処理バスに結合されて、そこからロードおよびストア命令を要取るためのロード／ストア機能ユニットとを含み、前記ロード／ストア機能ユニットは、前記内部アドレスデータ通信バスに結合されて、前記外部メモリに前記ロード／ストア機能ユニットアクセスを与え、さらに前記内部アドレスデータ通信バスおよび前記デコーダに結合されて、前記デコーダに命令源を与える命令キャッシュと、
前記内部アドレスデータ通信バスおよび前記ロード／ストア機能ユニットに結合されるデータキャッシュとをさらに含み、
前記内部アドレスデータ通信バスは、アドレスおよびデータ情報を前記外部メモリ、前記命令キャッシュおよび前記データキャッシュ間で通信する、請求項１に記載のマイクロプロセッサ。
【請求項１５】　命令およびデータを前記マイクロプロセッサに与えるための外部メモリと組合わされる、請求項１に記載のマイクロプロセッサ。
【請求項１６】　前記複数のオペランドバスが、オペランドおよびオペランドタグの両方がそれに伝達されるバスである、請求項６に記載のマイクロプロセッサ。
【請求項１７】　スーパースカラマイクロプロセッサであって、
同じマイクロプロセッササイクル内に複数の命令をデコードするための複数命令デコーダを含み、前記デコーダは、同じマイクロプロセッササイクル内に整数および浮動小数点命令の両方をデコードし、さらに前記デコーダに結合されるデータ処理バスと、
前記データ処理バスに結合される整数機能ユニットとを含み、前記整数機能ユニットは、前記マイクロプロセッサによる順序通りでない命令の実行を可能にするための複数の待合わせステーションを含み、さらに前記データ処理バスに結合される浮動小数点機能ユニットを含み、
前記浮動小数点機能ユニットは、前記マイクロプロセッ

サによる順序通りでない命令の実行を可能にするための複数の待合わせステーションを含み、さらに前記データ処理バスに結合されて、前記整数機能ユニットおよび前記浮動小数点機能ユニットの両方によって、そこから命令結果を受取って命令を推論的かつ順序通りではなく処理することを可能にするために用いられる共通リオーダバッファと、

前記リオーダバッファに結合されて、前記リオーダバッファから用済とされた命令結果を受入れるためのレジスタファイルと、

前記データ処理バスに結合されて、前記整数機能ユニットおよび浮動小数点機能ユニットの両方によって、コンピュータプログラム内のどの分岐が発生されるかを推論的に予測するために用いられる分岐予測ユニットと、

前記データ処理バスに結合されて、前記整数機能ユニットおよび前記浮動小数点機能ユニットの両方によって、情報のロードおよびストアを可能にするために用いられるロード／ストア機能ユニットとを含む、スーパースカラマイクロプロセッサ。

【請求項１８】　前記データ処理バスが、

複数のｏｐコードパスと、

複数のオペランドパスと、

複数の命令タイプパスと、

複数の結果パスと、

複数の結果タグパスとを含む、請求項１７に記載のマイクロプロセッサ。

【請求項１９】　前記オペランドパスがオペランドタグパスを含む、請求項１７に記載のマイクロプロセッサ。

【請求項２０】　前記データ処理バスが予め定められたデータ幅を示し、前記リオーダバッファが、前記データ処理バス幅に等しい幅を示すエントリと、前記データ処理バスのデータ幅の倍数に等しい幅を示すエントリとをストアするためのメモリ手段を含む、請求項１７に記載のマイクロプロセッサ。

【請求項２１】　前記デコーダが、プログラム順に整数および浮動小数点命令の両方を発行するための発行手段をさらに含む、請求項１７に記載のマイクロプロセッサ。

【請求項２２】　前記浮動小数点機能ユニットが、複数のサイズを示すオペランドを処理する、請求項１７に記載のマイクロプロセッサ。

【請求項２３】　前記浮動小数点機能ユニットが、単精度／倍精度浮動小数点機能ユニットを含む、請求項１７に記載のマイクロプロセッサ。

【請求項２４】　前記複数命令デコーダが、１マイクロプロセッササイクルにつき４つの命令をデコードすることができる、請求項１７に記載のマイクロプロセッサ。

【請求項２５】　前記マイクロプロセッサを、命令およびデータがストアされる外部メモリにインタフェースさせるための前記バスインタフェースユニットと、

前記バスインタフェースユニットに結合される内部アドレスデータ通信バスと、

前記内部アドレスデータ通信バスおよび前記デコーダに結合されて、前記デコーダに命令源を供給する命令キャッシュと、

前記内部アドレスデータ通信バスおよび前記ロード／ストア機能ユニットに結合されるデータキャッシュとをさらに含み、

前記内部アドレスデータ通信バスは、前記外部メモリ、前記命令キャッシュおよび前記データキャッシュ間でアドレスおよびデータ情報を通信する、請求項１７に記載のマイクロプロセッサ。

【請求項２６】　前記マイクロプロセッサに命令およびデータを与えるための外部メモリと組合わされる、請求項１７に記載のマイクロプロセッサ。

【請求項２７】　前記複数のオペランドバスが、オペランドおよびオペランドタグの両方がそれに伝達されるバスである、請求項１８に記載のマイクロプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の背景】この発明は一般にマイクロプロセッサに関し、より特定的には高性能スーパースカラマイクロプロセッサに関する。

【０００２】他の多くの近代技術分野と同様に、マイクロプロセッサの設計も、技術者および科学者が常に速度、効率および性能を高めようと努める技術である。一般的に言えば、マイクロプロセッサは２つのクラス、すなわちスカラおよびベクトルプロセッサに分けることができる。最も初期のスカラプロセッサは、１マシンサイクルにつき最大で１つの命令を処理する。いわゆる「スーパースカラ」プロセッサで、１マシンサイクルにつき処理できる命令は、１を上回る。スカラプロセッサと対照的に、ベクトルプロセッサは各マシンサイクル中に比較的大きな値のアレイを処理できる。

【０００３】ベクトルプロセッサは処理効率を追求するのにデータ並列性に頼り、一方スーパースカラプロセッサは動作の効率を高めるのに命令並列性に頼る。命令並列性は、命令を並列に処理することを可能にするこのような命令シーケンスの固有の特性と考えることができる。対照的に、データ並列性はその要素を並列に処理することを可能にするデータの流れの固有の特性と見ることができる。命令並列性は、命令の特定のシーケンスが示す従属性の数に関連する。従属性とは、ある特定の命令が別の命令の結果に依存する程度と定義される。スカラプロセッサでは、ある命令が別の命令に対する従属性を示すと、一般に、その命令が実行のために機能ユニットに渡され得る前にその従属性を解決しなくてはならない。この理由のため、従来のスカラプロセッサは、プロセッサがこのような従属性の未処理の解決を待つ間の望ましくない時間遅延がある。

【０００４】ここ数年、プロセッサおよびマイクロプロセッサによる命令の実行を高速化するためにいくつかのアプローチがとられてきた。現在でもマイクロプロセッサで広く用いられているアプローチの１つは、パイプライン化である。パイプライン処理では、１）命令のフェッチ、２）命令のデコードおよびオペランドの収集、ならびに３）命令の実行および結果のライトバックの３つのマイクロプロセッサの動作が処理を速くするために重ねられる、組立ラインのアプローチがとられる。言い換えれば、それぞれのマシンサイクルにおいて命令１がフェッチされ、命令１がデコードされる。命令１がデコードされ、そのオペランドが集められている間、命令２がフェッチされる。命令１が実行され、その結果が書込まれる間、命令２はデコードされ、そのオペランドが集められ、命令３がフェッチされる。実用において、組立ラインのアプローチは、上述したよりも多くの組立ラインステーションに分けられることがある。パイプライン技術のより詳細な議論は、ディー・ダブリュー・アンダーソン（D. W. Anderson）らによる、１９６７年１月、ＩＢＭジャーナル第１１巻の８－２４頁、「ＩＢＭシステム／３６０モデル９１：マシンフィロソフィ」（“The IBM System/360 Model 91:Machine Philosophy”）に記載される。

【０００５】以下の定義は、本明細書中、明確を期するために述べるものである。「発行」とは、命令を命令デコーダから機能ユニットに送る動作のことである。「投入」とは、命令を機能ユニット内での実行の状態に置く動作である。「完了」とは、命令が実行を終えて、その結果が利用可能であるときに達成されるものである。命令の結果がレジスタファイルに書込まれるとき、命令は「用尽」されると言う。これはまた、「ライトバック」とも称する。

【０００６】ウィリアム・ジョンソン（William Johnson）による最近の著書「スーパースカラマイクロプロセッサ設計」（“Superscalar Microprocessor Design”，１９９１年、プレンティス・ホール社（Prentice-Hall, Inc.））では、実用的なスーパースカラマイクロプロセッサの設計に関していくつかの一般的な考察が述べられている。図１は、このジョンソンの著書で説明されているスーパースカラマイクロプロセッサの実現例を示すマイクロプロセッサ１０のブロック図である。マイクロプロセッサ１０は、整数演算を処理するための整数ユニット１５と、浮動小数点演算を処理するための浮動小数点ユニット２０とを含む。整数ユニット１５および浮動小数点ユニットの各々は、それぞれ別個で専用の命令デコーダと、レジスタファイルと、リオーダバッファと、ロードおよびストアユニットとを含む。より特定的には、整数ユニット１５は、命令デコーダ２５と、レジスタファイル３０と、リオーダバッファ３５と、ロードおよびストアユニット（６０および６５）とを含み、一方浮動小数点ユニット２０は、固有の命令デコーダ４０と、レジスタファイル４５と、リオーダバッファ５０と、ロードおよびストアユニット（７５および８０）とを含み、図１に示されるとおりである。リオーダバッファはマイクロプロセッサの推論状態を含み、一方レジスタファイルはマイクロプロセッサのアーキテクチャの状態を含む。

【０００７】マイクロプロセッサ１０はメインメモリ５５に結合され、これは２つの部分、すなわち命令をストアするための命令メモリ５５Ａとデータをストアするためのデータメモリ５５Ｂとを含むものとして考えることができる。命令メモリ５５Ａは、整数ユニット１５と浮動小数点ユニット２０との両方に結合される。同様に、データメモリ５５Ｂも、整数ユニット１５および浮動小数点ユニット２０の両方に結合される。より詳細には、命令メモリ５５Ａはデコーダ２５およびデコーダ４０に命令キャッシュ５８を介して結合される。データメモリ５５Ｂは、データキャッシュ７０を介して整数ユニット１５のロード機能ユニット６０とストア機能ユニット６５とに結合される。データメモリ５５Ｂはまた、データキャッシュ７０を介して浮動小数点ユニット２０の浮動小数点ロード機能ユニット７５と浮動小数点ストア機能ユニット８０とに結合される。ロードユニット６０は、データメモリ５５Ｂから選択されたデータを整数ユニット１５へとロードする従来のマイクロプロセッサの機能を実行し、一方ストアユニット７０は、整数ユニット１５からのデータをデータメモリ５５Ｂにストアする従来のマイクロプロセッサの機能を実行する。

【０００８】コンピュータプログラムは、マイクロプロセッサ１０によって実行されるべき命令のシーケンスを含む。コンピュータプログラムは、典型的には、ハードディスク、フロッピィディスクまたはコンピュータシステム内に位置される他の不揮発性記憶媒体にストアされる。プログラムが実行されるとき、プログラムは記憶媒体からメインメモリ５５にロードされる。プログラムの命令および関連のデータが一旦メインメモリ５５内に入れば、個々の命令を実行のために準備し、最終的にはマイクロプロセッサ１０によって実行することができる。

【０００９】メインメモリ５５内にストアされた後、命令は、命令キャッシュ５８を介して命令デコーダ２５へと渡される。命令デコーダ２５は各命令を調べ、取るべき適切な動作を決定する。たとえば、デコーダ２５は、特定の命令が、ＰＵＣＨ、ＰＯＰ、ＬＯＡＤ、ＡＮＤ、ＯＲ、ＥＸ　ＯＲ、ＡＤＤ、ＳＵＢ、ＮＯＰ、ＪＵＭＰ、条件付ＪＵＭＰ（ＢＲＡＮＣＨ）または他のタイプの命令であるかを決定する。デコーダ５８が決定した特定のタイプの命令が存在するかに依存して、命令は適切な機能ユニットに発行される。ジョンソンの著書で提案されているスーパースカラアーキテクチャでは、デコーダ２５は１マシンサイクルにつき４つの命令をデコードすることのできるマルチ命令デコーダである。したがっ

て、デコード５８は４命令幅のバンド幅を示すと言える。

【００１０】図１に示されるように、ＯＰ　ＣＯＤＥバス８５は、デコード２５と機能ユニットの各々、すなわち分岐ユニット９０、算術論理装置９５および１００、シフタユニット１０５、ロードユニット６０およびストアユニット６５との間に結合される。この態様で、各命令のためのｏｐコードは適切な機能ユニットに与えられる。

【００１１】ここでしばらく直接的な説明からは離れるが、命令は、典型的には以下のフォーマットで、すなわちｏｐコード、オペランドＡ、オペランドＢ、行先レジスタという複数のフィールドを含むことが認められる。たとえば、サンプル命令ＡＤＤ　Ａ、Ｂ、Ｃとは、レジスタＡの内容をレジスタＢの内容に加算し、その結果を行先レジスタＣに置くことを意味するであろう。各命令のｏｐコード部分の処理は、既に上述したとおりである。ここで各命令のオペランドの処理を説明する。

【００１２】特定の命令のためのｏｐコードが適切な機能ユニットに送られなくてはならないだけでなく、その命令のための指定されたオペランドが検索されて、機能ユニットに送られなくてはならない。特定のオペランドの値がまだ計算されていなければ、機能ユニットが命令を実行できる前に、その値をまず計算して、機能ユニットに与えられなくてはならない。たとえば、現在の命令が先行の命令に従属していれば、現在の命令が実行される前に先行の命令の結果を決定しなくてはならない。この状況を従属性と称する。

【００１３】特定の命令を機能ユニットが実行するのに必要とされるオペランドは、レジスタファイル３０またはリオーダバッファ３５のいずれかによってオペランドバス１１０に与えられる。オペランドバス１１０は、機能ユニットの各々に結合される。したがって、オペランドバス１１０はオペランドを適切な機能ユニットに送る。実用において、オペランドバス１１０はオペランドＡおよびオペランドＢのための別個のバスを含む。

【００１４】機能ユニットにｏｐコードならびにオペランドＡおよびオペランドＢが与えられれば、機能ユニットは命令を実行し、その結果を、すべての機能ユニットの出力とリオーダバッファ３５とに（および、後述のように各機能ユニットの入力にあるそれぞれの待合わせステーションに）結合される結果バス１１５に置く。

【００１５】各機能ユニットの入力には、その命令のためのオペランドが機能ユニットに対してまだ利用可能でないという意味でまだ完全でない命令からのｏｐコードをストアするための「待合わせステーション」が設けられる。待合わせステーションは、後に待合わせステーションに到達する、抜けているオペランドのための場所を確保するオペランドタグとともに命令のｏｐコードをストアする。この技術は、未処理の命令が待合わせステーションでそのオペランドとともに集められている間、マイクロプロセッサが他の命令を実行し続けることを可能にすることによって性能を高める。図１に示されるように、分岐ユニット９０には待合わせステーション９０Ｒが設けられ、ＡＬＵ９５および１００には待合わせステーション９５Ｒおよび１００Ｒがそれぞれ設けられ、シフタユニット１０５には待合わせステーション１０５Ｒが設けられ、ロードユニット６０には待合わせステーション６０Ｒが設けられ、ストアユニット６５には待合わせステーション６５Ｒが設けられる。このアプローチでは、待合わせステーションが、より初期のマイクロプロセッサにおいて機能ユニットの入力で典型的には使用されていた入力ラッチの代わりに使用される。待合わせステーションに関してのよく知られた参考文献は、１９６７年１月、ＩＢＭジャーナル、第１１号、２５－３３頁、アール・エム・トマシュロ（Ｒ. Ｍ. Tomasulo）の「複数の算術装置を用いる効率的なアルゴリズム」（“An Efficient Algorithm For Expoiting Multiple Arithmetic Units”）である。

【００１６】先に述べたように、スカラマイクロプロセッサでの効果的なスループットを１マシンサイクルにつき１つの命令という限界まで増大するのにパイプラインを用いることができる。図１に示されるスーパースカラマイクロプロセッサでは、１マシンサイクルにつき複数の命令の処理を達成するのに複数のパイプラインが用いられる。この技術を、「スーパーパイプライン化」と称する。

【００１７】「レジスタ再指定」と称する別の技術もまた、スーパースカラマイクロプロセッサのスループットを高めるために用いることができる。この技術は、命令ストリームにおける２つの命令のどちらも同じレジスタ、たとえば仮説レジスタ１を使用することを要求する場合に有用である。第２の命令が第１の命令に従属していなければ、レジスタ１Ａと呼ぶ第２のレジスタが、レジスタ１の代わりに第２の命令によって使用されるように割当てられる。この態様で、レジスタ１を用いて第１の命令が終了するのを待つことなく、第２の命令を実行することができ、結果を得ることができる。図１に示されるスーパースカラマイクロプロセッサ１０は、命令処理能力を高めるのにレジスタ再指定のアプローチを用いる。マイクロプロセッサ１０においてレジスタ再指定を実現する態様を以下により詳細に説明する。

【００１８】上述のことから、レジスタ再指定がレジスタに対するストアの競合をなくすことが認められる。レジスタ再指定を実現するために、整数ユニット１５および浮動小数点ユニット２０は、それぞれのリオーダバッファ３５および５０と関連付けられる。簡略にするために、整数ユニット１５内のリオーダバッファ３５を介してのレジスタ再指定のみを議論するが、同じ議論が浮動小数点ユニット２０内の同様の回路にも当てはまる。

【0019】リオーダバッファ35は、命令結果にダイナミックに割当てられるいくつかのストア位置を含む。より特定的には、デコーダ25によって命令がデコードされると、その命令の結果値にリオーダバッファ35内の位置が割当てられ、その行先レジスタ番号がこの位置と関連付けられる。これが命令の行先レジスタ番号をリオーダバッファ位置に効果的に再指定する。タグ、または一時ハードウェア識別子が、結果を識別するためにマイクロプロセッサハードウェアによって発生される。このタグもまた、割当てられたリオーダバッファ位置にストアされる。レジスタにストアされていると考えられる値を得るために、命令ストリームにおける後の命令が再指定された行先レジスタを参照するとき、命令はその代わりにリオーダバッファにストアされた値、または値がまだ計算されていなければその値に関するタグを得る。

【0020】リオーダバッファ35は、内容参照メモリである、先入れ先出し（ＦＩＦＯ）環状バッファとして実現される。このことは、リオーダバッファ35内のエントリが、エントリを直接識別することによってではなく、エントリが含むものを特定することによって識別されることを意味する。より特定的には、エントリは、それに書込まれたレジスタ番号を用いて識別される。レジスタ番号がリオーダバッファ35に与えられると、リオーダバッファはレジスタに書込まれた最新の値（または値がまだ計算されていなければその値に関するタグ）を与える。このタグは、リオーダバッファ35内の特定の命令の相対的な進論位置を含む。この構成は、レジスタ番号を与えられるとレジスタ内の値を与えるレジスタファイル30を模倣している。しかしながら、リオーダバッファ35およびレジスタファイル30が用いる、その中の値にアクセスするための機構はかなり異なる。

【0021】リオーダバッファ35が用いる機構では、リオーダバッファは要求されたレジスタ番号をリオーダバッファのすべてのエントリ内のレジスタ番号と比較する。次に、リオーダバッファは一致するレジスタ番号を有するエントリの値（またはタグ）を戻す。これは連想ルックアップ技術である。対照的に、レジスタファイル30に要求されたレジスタ番号が与えられると、レジスタファイルは単にレジスタ番号をデコードし、選択されたエントリでの値を与える。

【0022】命令デコーダ25が命令をデコードすると、デコードされた命令のソースオペランドのレジスタ番号が、リオーダバッファ35およびレジスタファイル30に同時にアクセスするのに用いられる。リオーダバッファ35が、そのレジスタ番号が要求されたソースレジスタ番号と一致するエントリを持たない場合には、レジスタファイル30内の値がソースオペランドとして選択される。しかしながら、リオーダバッファ35が一致するエントリを有する場合には、そのエントリ内の値がソースオペランドとして選択される、というのはこの値はリオーダバッファに割当てられた最も最近の値であるはずだからである。値がまだ計算されていないために利用可能でなければ、その値に関するタグがその代わりに選択され、オペランドとして用いられる。いずれの場合にせよ、値またはタグが適切な機能ユニットの待合わせステーションにコピーされる。この手順が、デコードされた命令の各々が要求する各オペランドについて行なわれる。

【0023】典型的な命令シーケンスでは、所与のレジスタは何度も書込まれる。この理由のため、命令が同じレジスタを特定する場合には、それらの命令によって同じレジスタがリオーダバッファ35の異なるエントリに書込まれる可能性がある。この状況で正しいレジスタ値を得るために、リオーダバッファ35は割当の順番によって複数の一致エントリに優先順位をつけ、特定のレジスタ値が要求されると最も最近のエントリを戻す。この技術によって、リオーダバッファへの新しいエントリが、より古いエントリにとって替わる。

【0024】機能ユニットが結果を生成すると、その結果はリオーダバッファ35、およびその結果に関するタグを含む何らかの待合わせステーションのエントリに書込まれる。結果値がこの態様で待合わせステーションに書込まれると、必要なオペランドを与えるかもしれず、実行のために機能ユニットに投入されるべき1つまたはそれ以上の待合わせをしている命令を解放するかもしれない。結果値がリオーダバッファ35に書込まれた後、後続の命令はリオーダバッファから結果値をフェッチし続ける。このフェッチングは、エントリが新しい値にとって替わられなければ、かつ、値をレジスタファイル30に書込むことによって値が用済とされるまで続く。用尽は、元の命令シーケンスの順序で起こり、したがって割込および例外に関して順序通りの状態を保つ。

【0025】浮動小数点ユニット20に関しては、浮動小数点ロード機能ユニット75および浮動小数点ストア機能ユニット80に加えて、浮動小数点ユニット20は他の機能ユニットも含むことがわかる。たとえば、浮動小数点ユニット20は、浮動小数点加算ユニット120と、浮動小数点変換ユニット125と、浮動小数点乗算ユニット130と、浮動小数点除算ユニット140とを含む。OP　CODEバス145が、デコーダ40と浮動小数点ユニット20内の各機能ユニットとの間に結合されて、デコードされた命令を機能ユニットに与える。各機能ユニットはそれぞれの待合わせステーション、すなわち浮動小数点加算待合わせステーション120Rと、浮動小数点変換待合わせステーション125Rと、浮動小数点乗算待合わせステーション130Rと、浮動小数点除算待合わせステーション140Rとを含む。オペランドバス150は、レジスタファイル45およびリオーダバッファ50を機能ユニットの待合わせステーションに結合して、オペランドがそれらに与えられるよう

にする。結果バス１５５は、浮動小数点ユニット２０のすべての機能ユニットの出力をリオーダバッファ５０に結合する。リオーダバッファ５０はレジスタファイル４５に結合される。リオーダバッファ５０およびレジスタファイル４５には、したがって、先に整数ユニット１５に関して説明したのと同じ態様で結果が与えられる。

【００２６】整数リオーダバッファ３５は１６のエントリを保持し、浮動小数点リオーダバッファ５０は８のエントリを保持する。整数リオーダバッファ３５および浮動小数点リオーダバッファ５０は、各々１マシンサイクルにつき２つの計算値を受入れることができ、１サイクルにつき２つの結果をそれぞれのレジスタファイルに格納することができる。

【００２７】マイクロプロセッサがデコードされた命令を順序通りに投入する（「順序通りの投入」）ように制約されると、マイクロプロセッサは、デコードされた命令が資源の競合を発生する（すなわち２つの命令の両方がＲ１レジスタを使うことを要求する）と常に、またはデコードされた命令が従属性を有すると、命令のデコードを停止しなくてはならない。対照的に、「順序通りでない投入」を用いる図１のマイクロプロセッサ１０は、デコーダ２５を実行ユニット（機能ユニット）から分離することによって、このタイプの命令の投入を達成する。これは、リオーダバッファ３５および機能ユニットにある上述の待合わせステーションを用いて分配命令ウィンドウを効果的に確立することによって行なわれる。この態様で、デコーダは、命令を直ちに実行できなくても、命令をデコードし続けることができる。命令ウィンドウは、マイクロプロセッサが、先に進み命令を実行し続けながらそこから引出すことのできる命令のプールとして作用する。したがって、命令ウィンドウによってマイクロプロセッサに先見能力が与えられる。従属性がクリアされてオペランドが利用可能になると、ウィンドウ内のより多くの命令が機能ユニットによって実行され、デコーダはさらに多くのデコードされた命令でウィンドウを充満し続ける。

【００２８】マイクロプロセッサ１０は、その性能を高めるために分岐予測ユニット９０を含む。プログラムの命令ストリームにおける分岐がマイクロプロセッサの命令をフェッチする能力を妨げることはよく知られている。これは、分岐が起こると、フェッチャがフェッチすべき次の命令が分岐の結果に従属するからである。ユニット９０等の分岐予測ユニットがなければ、マイクロプロセッサの命令フェッチャは機能停止となるか、または正しくない命令をフェッチする恐れがある。このことは、マイクロプロセッサが命令ウィンドウ内の並列に実行する他の命令を探しあてる可能性を減じてしまう。ソフトウェア分岐予測ではなく、ハードウェア分岐予測が分岐予測ユニット９０では用いられて、命令のフェッチの間に起こる分岐の結果を予測する。言い換えれば、分岐予測ユニット９０は、分岐が発生されるべきであるか否かを予測する。たとえば、先行の分岐結果の実行の履歴を保持するために分岐先バッファが用いられる。この履歴に基づいて、特定のフェッチされた分岐の間、フェッチされた分岐命令がどの分岐をとるかに関して決定がなされる。

【００２９】ソフトウェア分岐予測もまた、分岐の結果を予測するのに用いることができることが認められる。この分岐予測のアプローチでは、プログラムにおける各分岐にいくつかのテストが行なわれて、統計的にどの分岐結果が起こりそうかを判断する。ソフトウェア分岐予測技術は、典型的にはプログラム自体に好ましい分岐結果に関して統計的な分岐予測情報を組込むことを伴う。コード列（分岐等）が、マイクロプロセッサがそのコード列を実行するのが適切であることを確信する前に実行されるマイクロプロセッサ設計の実用に、「推論実行」という用語がしばしば適用される。

【００３０】スーパースカラマイクロプロセッサの動作を理解するために、パイプラインの各ステージ、すなわちフェッチ、デコード、実行、ライトバックおよび結果コミットでのスカラおよびスーパースカラマイクロプロセッサを比較することが有用である。以下の表１はこのような比較を示す。

【００３１】

【表１】

パイプライン段階	パイプライン化されたスカラプロセッサ	パイプライン化されたスーパースカラプロセッサ（投入および完了は順序通りでない）
フェッチ	1つの命令をフェッチする	複数の命令をフェッチする
デコード	命令をデコードする レジスタファイルからオペランドにアクセスする 機能ユニット入力ラッチにオペランドをコピーする	命令をデコードする レジスタファイルおよびリオーダバッファからオペランドにアクセスする 機能ユニット待合わせステーションにオペランドをコピーする
実行	命令を実行する	命令を実行する 結果バスに対して調停する
ライトバック	レジスタファイルに結果を書込む 機能ユニット入力ラッチに結果を転送する	リオーダバッファに結果を書込む 結果を機能ユニットの待合わせステーションに転送する
結果コミット	n／a	レジスタファイルに結果を書込む

【００３２】スーパースカラマイクロプロセッサ１０の上述の説明より、このマイクロプロセッサは実に強力であるが、非常に複雑な構造であることが認められる。しかしながら、設計の簡略化および処理性能のさらなる向上が、マイクロプロセッサ１０等のマイクロプロセッサにおいて常に望ましい。

【００３３】

【発明の概要】したがって、本発明のスーパースカラマイクロプロセッサのある利点は、並列に命令を処理することに関しての性能の向上である。

【００３４】本発明のスーパースカラマイクロプロセッサの別の利点は、その複雑さが減じられたことである。

【００３５】本発明のスーパースカラマイクロプロセッサのさらに別の利点は、他のスーパースカラマイクロプロセッサと比較して、ダイの寸法が減じられたことである。

【００３６】本発明の一実施例に従えば、主メモリにストアされた命令を処理するためのスーパースカラマイクロプロセッサが提供される。マイクロプロセッサは、同じマイクロプロセッササイクル内に複数の命令をデコードするための複数命令デコーダを含む。デコーダは、同じマイクロプロセッサ内に整数および浮動小数点命令の両方をデコードする。マイクロプロセッサは、デコーダに結合されるデータ処理バスを含む。マイクロプロセッサはさらに、同じデータ処理バスに結合されて、これを共有する整数機能ユニットおよび浮動小数点機能ユニットを含む。共通のリオーダバッファが、データ処理バスに結合されて、整数機能ユニットおよび浮動小数点機能ユニットの両方に用いられる。共通レジスタファイルがリオーダバッファに結合されて、リオーダバッファから用済とされた命令結果を受入れる。

【００３７】新規であると考えられる本発明の特徴は、前掲の特許請求の範囲に特定的に述べられる。しかしながら、この発明自体は、その構造および動作方法の両方について、以下の説明および添付の図面を参照することによって最もよく理解されるであろう。

【００３８】

【実施例の詳細な説明】

１．スーパースカラマイクロプロセッサ概説

本発明の高性能スーパースカラマイクロプロセッサは、望ましいことに、順序通りでない命令の投入と順序通りでない命令の実行とを並列して可能にする。より特定的には、開示されるスーパースカラマイクロプロセッサでは、命令はプログラム順に発行され、投入および完了は順序通りでなく、用呂（用済）は順序通りに行なわれる。高性能を可能にする本発明のいくつかの局面を、より詳細な説明に入る前に議論する。

【００３９】図２のスーパースカラマイクロプロセッサ２００は、いくつかの主な構成要素を共有することで、ダイの寸法を増大することなく性能を向上することができる。このマイクロプロセッサのアーキテクチャでは、整数ユニット２１５および浮動小数点ユニット２２５は共通のデータ処理バス５３５に結合される。データ処理バス５３５は、主にその広いバンド幅のために、高速で高性能のバスである。整数機能ユニットおよび浮動小数点機能ユニットが別個のバスの上にある設計と比較して、これらの両方の機能ユニットをさらに活用することが可能になる。

【００４０】整数および浮動小数点機能ユニットは、複数の待合わせステーションを含み、これらもまた同じデータ処理バス５３５に結合される。図３ないし図５に示される本発明のマイクロプロセッサのより詳細な表現からわかるように、整数および浮動小数点機能ユニットはまた、データ処理バス５３５を介して共通の分岐ユニッ

ト５２０を共有する。さらに、整数および浮動小数点機能ユニットは、同じデータ処理バス５３５に結合される共通のロード／ストアユニット５３０を共有する。開示されるマイクロプロセッサアーキテクチャは、マイクロプロセッサダイの寸法をより効率的に用いながら、有利に性能を高める。図２ないし図５に示されるこの発明の実施例では、本発明のマイクロプロセッサは、マイクロプロセッサによって処理される命令が同じ幅を示し、かつオペランドサイズが可変である縮小命令セットコンピュータ（ＲＩＳＣ）である。

【００４１】図２に戻って、この発明のスーパースカラマイクロプロセッサの簡略化されたブロック図が、マイクロプロセッサ２００として示される。スーパースカラマイクロプロセッサ２００は、４命令幅、２ウェイセットアソシアティブ、部分デコード８Ｋバイト命令キャッシュ２０５を含む。命令キャッシュ２０５は、分岐予測を伴う１マシンサイクルにつき複数の命令のフェッチをサポートする。この明細書の目的のため、マシンサイクルおよびマイクロプロセッササイクルという用語は、同意語であると見なす。命令キャッシュ２０５はまた、ＩＣＡＣＨＥとも称する。

【００４２】マイクロプロセッサ２００はさらに、オペランドの利用可能性に関わらず、１マシンサイクルにつき４つまでの命令をデコードし、６つの独立した機能ユニットのいずれにも発行することのできる命令デコーダ（ＩＤＥＣＯＤＥ）２１０を含む。図３ないし図５にマイクロプロセッサ５００として示される本発明のより詳細な実施例においてわかるように、これらの機能ユニットは、２つの算術論理ユニット（まとめてＡＬＵ５００として示されるＡＬＵ０およびＡＬＵ１）を含む。これらの機能ユニットはさらに、シフタセクション５１０（ＳＨＦＳＥＣ）を含み、これはＡＬＵセクション５０５とともに、整数命令を処理するための整数ユニット５１５を形成する。機能ユニットはさらに、命令分岐を処理し、かつ分岐予測を行なうための分岐セクション（ＢＲＮＳＥＣ）５２０を含む。分岐ユニット５２０として用いることができる分岐ユニットの１つは、１９９２年８月４日に発行された、「キャッシュ内に各命令のブロックとストアされたフェッチ情報を用いての適切に予測された分岐命令に続く実行のための遅延を低減するためのシステム」（“System For Reducing Delay For Execution Subsequent ToCorrectly Predicted Branch Instruction Using Fetch Information Stored With Each Block Of Instructions In Cache”）と題される米国特許第５，１３６，６９７号に記載され、その開示をここに引用によって採用する。浮動小数点セクション（ＦＰＴＳＥＣ）５２５およびロード／ストアセクション（ＬＳＳＥＣ）５３０もまた、デコーダ（ＩＤＥＣＯＤＥ）２１０が命令を発行する機能ユニットに含まれる。上述の機能ユニットはすべて、図３ないし図５に示されるように共通の主データ処理バス５３５を共有する（この明細書の目的のため、図３ないし図５は併せてマイクロプロセッサ５００を形成し、併せて横に並べて見るものである）。

【００４３】図２のスーパースカラマイクロプロセッサ２００の簡略化されたブロック図では、分岐は整数演算と考えられ、分岐ユニットは整数コア２１５の一部として見なされる。スーパースカラマイクロプロセッサ２００は、オペランド従属性の適切な順序付けを守り、かつ順序通りでない投入を可能にするために命令のタグの付与を行なう。マイクロプロセッサ２００はさらに、発行された命令が実行を待つ間待ち行列にされる、機能ユニットの複数の待合わせステーションを含む。この特定の実施例では、各機能ユニットの入力に２つの待合わせステーションが設けられる。より特定的には、この特定の実施例では、整数コア２１５は２つの待合わせステーション２２０を含み、浮動小数点コア２２５は２つの待合わせステーション２３０を含む。１機能ユニットについて用いられる待合わせステーションの数は、所望される待ち行列の程度に従って変えてもよい。整数コア２１５は整数命令を処理し、浮動小数点コア２２５は浮動小数点命令を処理する。実用において、整数コア２１５および浮動小数点コア２２５の各々は、複数の機能ユニットを含み、この発明の一実施例では、その各々には複数の待合わせステーションが備えられる。

【００４４】この特定の実施例において、マイクロプロセッサ２００は１マシンサイクルについて３つまでの機能ユニット結果を処理することができる。これは、マイクロプロセッサ２００が、すべての機能ユニット（すなわち図２の整数コア２２０および浮動小数点コア２３０）に結合されるＲＥＳＵＬＴ０、ＲＥＳＵＬＴ１、およびＲＥＳＵＬＴ２とされる３つの結果バスを含むからである。この発明はこの数の結果バスに制限されるわけではなく、所望の性能レベルに見合った、より多いまたは少ない数の結果バスを用いてもよい。同様に、この発明は示される実施例における機能ユニットの特定の数に制限されるわけではない。

【００４５】マイクロプロセッサ２００はさらに、リオーダバッファ２４０から用済となった結果をストアするための統合されたレジスタファイル２３５を含む。レジスタファイル２３５は、一実施例においては１マシンサイクルにつき４つの書込および２つの書込を可能にするマルチポートマルチレジスタ記憶領域である。レジスタファイル２３５は様々なサイズのエントリ、すなわち一実施例では同じレジスタファイルに３２ビット整数および６４ビット浮動小数点オペランドエントリの両方を収容する。レジスタファイル２３５は、この特定の実施例では１９４の３２ビットレジスタのサイズを示す。リオーダバッファ２４０もまた異なるサイズのエントリ、すなわち一実施例では同じレジスタファイル内に３２ビッ

ト整数および64ビット浮動小数点オペランドエントリの両方を収容する。これらの特定の数もまた、制限するものではなく例示する目的のために与えるものである。

【0046】リオーダバッファ240は、環状バッファ、または順序通りでない機能ユニットの結果を受取りかつ逐次命令プログラム順にレジスタファイル235を更新するキューである。一実施例では、リオーダバッファ240は、10のエントリを備えた先入れ先出し（FIFO）バッファとして実現される。FIFO ROB240内のキューは先頭および末尾を含む。この発明の別の実施例では、16のエントリを備えたリオーダバッファを用いる。リオーダバッファ240は再指定されたレジスタに割当てられる位置を含み、推論的に実行された命令の結果を保持する。分岐論理がある分岐の発生を予測すると、予測された分岐における命令が、分岐がある特定の例において適切に発生したとの推論の下に実行されるように、命令が推論的に実行される。分岐が誤予測されたと判断されるようなことがあれば、リオーダバッファ240内にある分岐結果は、効果的にキャンセルされる。このことは、マイクロプロセッサが誤予測された分岐命令に対して効果的にバックアップし、マイクロプロセッサの推論状態をリセットし、誤予測された分岐前のプログラム命令ストリームの点から実行を再開することによって達成される。

【0047】リオーダバッファの10のエントリは各々32ビット幅（32ビット整数量の幅に対応する）であるが、リオーダバッファはまた、たとえば64ビット浮動小数点量等の64ビット量を収容することもできる。これは、リオーダバッファ内で64ビット量を2つの連続ROPとしてストアすることによって達成される（アール・オップと発音するROPは、マイクロプロセッサによって処理されるRISCまたはRISC類似命令／演算を指す）。このようにストアされた連続ROPは、これらを1つの構造として連結する情報を有し、1つの構造として一緒に用済とされる。各リオーダバッファエントリは、1の32ビット量、すなわち倍精度浮動小数点量の1／2、1の単精度浮動小数点量または32ビット整数を保持する容量を有する。

【0048】プログラムカウンタ（PC）は、もう推論的ではないものとしてレジスタファイル235に格納された命令と、推論的に実行されてその結果がリオーダバッファ（ROB）240にあり、用済が未定の命令との間の境界である、プログラム命令ストリーム内の点を追跡するために用いられる。このPCは、リタイアPCまたは単にPCと称する。リタイアPCは、ROBキューの先頭にストアされ、更新される。ROBエントリは、相対PC更新状態情報を含む。

【0049】リタイアPCは、リオーダバッファキューの先頭と関連する状態情報によって更新される。より特定的には、リオーダバッファキューは、この特定の実施例では最大4の命令までの、用済とする準備のできている命令の数を示す。リタイア論理242内に位置されるリタイアPCセクションは、現在の用済となったPCを保持する。ある特定のクロックサイクル内に4つの逐次命令が用済とされるべきであれば、リタイアPC論理は現在のリタイアPCに［4命令＊4バイト／命令］を加えて新しいリタイアPCを生成する。発生された分岐が存在すれば、リタイアPCは、一旦分岐が用済とされもう推論的でなくなると、分岐先に進む。リタイアPCは次に、その点から用済とされた命令の数だけ増分される。リタイアPCはリタイア論理242内の内部バス、すなわちPC（31：0）に存在する。

【0050】II．スーパースカラマイクロプロセッサの簡略化されたブロック図

このセクションでは、図2の簡略化されたマイクロプロセッサのブロック図のまだ述べていない局面を中心に議論する。一般的な見方を述べる。

【0051】図2は、マイクロプロセッサ200として、この発明の高性能スーパースカラマイクロプロセッサの一実施例の簡略化されたブロック図を示す。マイクロプロセッサ200において、命令キャッシュ205およびデータキャッシュ245は、32ビット幅内部アドレスデータ（IAD）バス250を介して互いに結合される。IADバス250は、一実施例では、データ処理バス535と比較すると比較的低速の通信バスである。IADバス250は、マイクロプロセッサ200のいくつかの主要な構成要素を相互接続して、このような構成要素の間でアドレス情報およびデータの両方の通信を与えるように機能する。IADバス250は、データ処理バス535が扱うオペランド処理および結果処理のように高速の並列性を要求しないタスクのために用いられる。この発明の一実施例では、IADバス250は、各クロックサイクルにおいてデータおよびアドレス情報の両方がそれにマルチプレクスされる32ビット幅バスである。IADバス250のバンド幅は、したがって高例では64ビット／クロックである。

【0052】主メモリ255が、図2に示されるようにバスインタフェースユニット260を介してIADバス250に結合される。このように、主メモリ255へのおよびそこからの情報の読出および書込が可能にされる。図示の目的のため、主メモリ255はマイクロプロセッサ200の一部として図2に示される。実用において、主メモリ225は、一般にマイクロプロセッサ200の外部に置かれる。

【0053】しかしながら、たとえばマイクロコントローラの場合のように主メモリ255がマイクロプロセッサ200内に配置される、マイクロプロセッサ200の実現例が企図される。

【0054】デコーダ210は、命令キャッシュ205に結合されるフェッチャ257を含む。フェッチャ25

7は、デコーダ210によるデコードおよび発行のためにキャッシュ205および主メモリ255から命令をフェッチする。

【0055】バスインタフェースユニット（BIU）260は、IADバス250に結合されてマイクロプロセッサ200の外部にあるバス回路（図示せず）とマイクロプロセッサ200をインタフェースさせる。より特定的には、BIUバス260は、マイクロプロセッサ200の外部にあるシステムバス、ローカルバスまたは他のバス（図示せず）とマイクロプロセッサ200をインタフェースさせる。BIU260として用いることができるバスインタフェースユニットの1つは、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッド（Advanced Micro Devices）が製造するAM29030マイクロプロセッサからのバスインタフェースユニットである。BIU260は、A（31：0）と示されるアドレスポートと、D（31：0）と示されるデータポートとを含む。BIU260はまた、バスハンドシェークポート（BUS HAND SHAKE）と、XBREQ（バスリクエストなし）およびXBGRT（バスグラントなし）と示されるグラント／リクエストラインとを含む。AM29030マイクロプロセッサのバスインタフェースユニットは、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドの出版するAm29030ユーザーズマニュアルにより詳細に説明される。

【0056】当業者には、命令列およびそのためのデータを含むプログラムが主メモリ255にストアされることが認められるであろう。命令およびデータがメモリ255から読出されると、命令およびデータは、命令がデコーダ210によってフェッチされ、デコードされ、機能ユニットに発行され得る前に、それぞれ命令キャッシュ205およびデータキャッシュ245にストアされる。

【0057】デコーダ210によって特定の命令がデコードされると、デコーダ210はデコードされた命令のopコードをその命令のタイプのための適切な機能ユニットに送る。たとえば以下の命令、すなわちADD R1，R2，R3（レジスタ1内の整数をレジスタ2内の整数に加えてその結果をレジスタ3に置く）がフェッチされたと仮定する。ここで、R1はAオペランドであり、R2はBオペランドであり、R3は行先レジスタである。

【0058】実用において、デコーダ210は1度に1ブロックにつき4つの命令をデコードし、各命令に関連するopコードを識別する。言い換えれば、デコード210は、デコード210に含まれる4つの発行位置の各々のためのopコードタイプを識別する。4つのデコードされたopコードタイプは、それぞれ4つのTYPEバスを介して機能ユニットにブロードキャストされる。4つのデコードされたopコードはそれぞれのOP CODEバスを介して機能ユニットにブロードキャストされる。もし利用可能であれば、オペランドがROB240およびレジスタファイル235から検索される。オペランドは、Aオペランドおよびオペランドバスを介して機能ユニットにブロードキャストされる。特定のオペランドが利用可能でなければ、AおよびBオペランドタグがその代わりに適切なAまたはBオペランドバスを介して適切な機能ユニットに送られる。デコーダ210によってデコードされた4つの命令は、このように処理のために機能ユニットに発行される。

【0059】この例でのADD opコードに関して、機能ユニットの1つ、すなわち整数コア215内の算術論理装置（ALU）は、opコードタイプを認め、その待合わせステーション220においてopコード、Aオペランドタグ、Aオペランド（もし利用可能であれば）、Bオペランドタグ、Bオペランド（もし利用可能であれば）および行先タグを含む情報をラッチする。ALU機能ユニットは次に結果を判断し、その結果を、ROB240でのストアのために、および未処理の命令を処理するためにその結果を必要としている何らかの他の機能ユニットによる検索のために、結果バス265に置く。

【0060】命令がデコーダ210によってデコードされると、その結果のためにリオーダバッファ240内のレジスタが割当てられることが認められる。次に命令の行先レジスタが、割当てられたレジスタと関連付けられる。命令はまだ利用可能でない結果に対応する結果タグ（一時の一意的ハードウェア識別子）が割当てられたレジスタに置かれる。「レジスタ再指定」がこのように実現される。プログラム命令列における後の命令が、リオーダバッファ240内のこの再指定された行先レジスタを参照すると、リオーダバッファ240は、そのレジスタに割当てられた位置にストアされた結果値か、またはその結果がまだ計算されていなければその値のためのタグのいずれかを与える。結果が計算されると、結果タグバスに信号が与えられ、リオーダバッファ240および機能ユニットの待合わせステーションに結果バスを介して結果が利用可能となったことを知らせる。このようにして結果がリオーダバッファ240にストアされる。

【0061】図3および4に示されるように、行先タグラインはリオーダバッファ240から機能ユニットに延びる。デコーダ210は、リオーダバッファに、リオーダバッファエントリの割当の準備が現在できている命令の数を知らせる。リオーダバッファは次に、リオーダバッファの現在の状態に基づいて行先タグを各命令に割当てる。デコーダ210は次に、各命令が投入されるか否かを確立する。リオーダバッファは投入された命令を取込み、リオーダバッファエントリの一時的割当を確立する。

【0062】特定の命令のためのオペランドは、共通デ

ータ処理バス５３５のＡオペランドバス（Ａ　ＯＰＥＲ）およびＢオペランドバス（Ｂ　ＯＰＥＲ）を介して、適切な機能ユニットに送られる。それぞれの命令の結果は、これらの命令に割当てられた機能ユニットで発生する。これらの結果は、３つの結果バスＲＥＳＵＬＴ０、ＲＥＳＵＬＴ１およびＲＥＳＵＬＴ２を含む複合結果バス２６５を介してリオーダバッファ２４０に送られる。複合結果バス２６５は、データ処理バス５３５の一部である。

【００６３】特定の命令がデコードされたときに、１つまたはそれ以上のオペランドが現在利用可能でないことは、デコーダ２１０から機能ユニットへの命令の発行を妨げるわけではない。そうではなく、１つまたはそれ以上のオペランドがまだ利用可能でない場合には、オペランドタグ（一時の一意的ハードウェア識別子）が、抜けているオペランドの代わりに適切な機能ユニット／待合わせステーションに送られる。オペランドタグおよび命令のためのｏｐコードは、タグに対応するオペランドが結果バスを介してリオーダバッファ２４０で利用可能となるまでは、その機能ユニットの待合わせステーションにストアされる。抜けていたすべてのオペランドがリオーダバッファ２４０で利用可能となれば、タグに対応するオペランドがリオーダバッファ２４０から検索される。オペランドおよびｏｐコードは、待合わせステーションから実行のために機能ユニットに送られる。結果は、リオーダバッファ２４０に伝送するために結果バスに置かれる。

【００６４】上述のオペランドタグトランザクションにおいて、Ａ　ＯＰＥＲおよびＢ　ＯＰＥＲバスを介して機能ユニットの待合わせステーションにオペランドタグが実際に送られることが認められる。オペランドタグをやりとりするためにこのような態様で用いられると、Ａ　ＯＰＥＲおよびＢ　ＯＰＥＲバスは、図２に示されるようにＡ　ＴＡＧおよびＢ　ＴＡＧと称する。

【００６５】ＩＩＩ．スーパースカラマイクロプロセッサ：より詳細な説明

図３ないし図５は、マイクロプロセッサ５００として、この発明のマイクロプロセッサのより詳しい実現例を示す。図２ないし図５に示されるマイクロプロセッサ内の同様の要素を示すのに同様の参照符号を用いる。マイクロプロセッサ５００のある部分は既に説明したことが認められる。

【００６６】マイクロプロセッサ５００において、命令は推論プログラム順に発行され、投入および完了は順番通りではなく、順番通りに用済とされる。多くの信号およびバスが、特に命令の発行に関して並列性を促進するために複製されることが後の説明より明らかになるであろう。デコーダ２１０は、１マイクロプロセッササイクルについて複数の命令をデコードし、デコードされた命令がそこから機能ユニットに並列に発行される発行ウィンドウを形成する。ＩＣＡＣＨＥ２０５は、１度に４つの命令をデコーダ２１０に、ＩＣＡＣＨＥ２０５をデコーダ２１０に結合するラインＩＮＳ０、ＩＮＳ１、ＩＮＳ２およびＩＮＳ３を介して与えることができる。

【００６７】マイクロプロセッサ５００において、主データ処理バスは、やはりデータ処理バス５３５として示される。データ処理バス５３５は４つのＯＰ　ＣＯＤＥバスと、４つのＡ　ＯＰＥＲ／Ａ　ＴＡＧバスと、４つのＢ　ＯＰＥＲ／Ｂ　ＴＡＧバスと、４つのＯＰ　ＣＯＤＥ　ＴＹＰＥバスとを含む。４つのＯＰ　ＣＯＤＥバス、４つのＡ　ＯＰＥＲ／Ａ　ＴＡＧバス、４つのＢ　ＯＰＥＲ／Ｂ　ＴＡＧバス、および４つのＯＰ　ＣＯＤＥ　ＴＹＰＥバスは、デコードされた命令を機能ユニットに伝送するように協働するため、これらは併せて、ＸＩＯＢ、ＸＩ１Ｂ、ＸＩ２ＢおよびＸＩ３Ｂ（図では別個に符号を付けられるわけではない）と示される４つの命令バスとしても参照される。これらの煩似した命令バスの名称は、互いから１桁で区別される。この桁は０をより早い命令として、０ｍｏｄ１６バイトメモリブロックにおける命令の位置を示す。これらの名称はここでは小文字「ｎ」でその桁を示す一般的な形で与えられる（すなわち、４つの命令バスＸＩＯＢ、ＸＩ１Ｂ、ＸＩ２ＢおよびＸＩ３Ｂは、ＸＩｎＢとして参照する）。

【００６８】順序通りでない命令の並列の実行を可能にするスーパースカラマイクロプロセッサ５００の特徴を、ここでマイクロプロセッサのより詳細な説明を始める前に簡単に繰返す。マイクロプロセッサ５００は、４命令幅、２ウェイセットアソシアティブ、部分デコード８Ｋバイト命令キャッシュ２０５（ＩＣＡＣＨＥ）を含み、分岐予測を伴う、１マイクロプロセッササイクルにつき４つの命令のフェッチをサポートする。マイクロプロセッサ５００は、オペランドの利用可能性に関わらず、５つの独立した機能ユニットのうちの何らかのものへのデコーダ２１０（ＩＤＥＣＯＤＥ）による１サイクルにつき４つまでの命令のデコードおよび発行を与える。これらの機能ユニットは、分岐セクションＢＲＮＳＥＣ５２０、算術論理装置ＡＬＵ５０５、シフタセクションＳＨＦＳＥＣ５１０、浮動小数点セクションＦＰＴＳＥＣ５２５、およびＬＯＡＤ／ＳＴＯＲＥセクション５３０を含む。

【００６９】マイクロプロセッサ５００は、オペランドの促属性の適切な順序付けを守り、順序通りでない投入を可能にするために、命令のタグ付与を行なう。マイクロプロセッサ５００はさらに、まだ実行できない発行された命令がそこで待ち行列にされる、機能ユニット内の待合わせステーションを含む。３つの結果バス（ＲＥＳＵＬＴ０、ＲＥＳＵＬＴ１およびＲＥＳＵＬＴ２）が、１サイクルにつき３つまでの機能ユニット結果を扱うことを可能にするように設けられる。環状バッファまたはＦＩＦＯキュー、すなわちリオーダバッファ２４０が、

順序通りでない機能ユニットの結果を受取り、レジスタファイル２３５を更新する。より特定的には、レジスタファイルはリオーダバッファからの結果で正しいプログラム順に更新される。言い換えれば、リオーダバッファからレジスタファイルへの結果の格納は、それが関係するすべての分岐、算術およびロード／ストア動作とともに正しい実行順に行なわれる。マルチポートレジスタファイル２３５は、１マシンサイクルにつき４つの読出および２つの書込ができる。ＲＥＳＵＬＴ０、ＲＥＳＵＬＴ１およびＲＥＳＵＬＴ２は、ＲＯＢ２４０に並列に書込まれる。結果がＲＯＢ２４０から用済とされる際、これらは書込バスＷＲＩＴＥＢＡＣＫ０およびＷＲＩＴＥＢＡＣＫ１を介して並列にレジスタファイル２３５に書込まれる。マイクロプロセッサ５００はまた、ロードおよびストア待ち時間を最少にするように、オンボードのダイレクトマッピング８Ｋバイトコヒーレントデータキャッシュ２４５を含む。

【００７０】［ＩＩＩ（ａ）命令フロー－フェッチ］マイクロプロセッサ５００の命令フローをここで説明する。命令デコーダ（ＩＤＥＣＯＤＥ）２１０は、命令を命令キャッシュ（ＩＣＡＣＨＥ）２０５からフェッチする命令フェッチャ２５７を含む。キャッシュ２０５として用いることができる命令キャッシュの１つは、１９９２年４月１２日に出願された、「命令デコーダおよびこれを用いるスーパースカラプロセッサ」（"Instruction Decoder And Superscalar Processor Utilizing Same"）と題される同時係属中の米国特許出願連続番号第０７／９２９，７７０号に説明され、本明細書においてこれを引用によって援用する。デコーダ２１０（ＩＤＥＣＯＤＥ）として用いることができるデコーダの１つもまた、１９９２年４月１２日に出願された「命令デコーダおよびこれを用いるスーパースカラプロセッサ」と題される米国特許出願連続番号第０７／９２９，７７０号に説明される。

【００７１】主メモリ２５５内の特定のプログラムがマイクロプロセッサ５００によって実行されるとき、プログラムの命令は実行のためにプログラム順に検索される。命令は通常最初はＩＣＡＣＨＥ２０５にないので、まず典型的なＩＣＡＣＨＥリフィル動作を説明する。キャッシュミスの際に、０ｍｏｄ１６バイト（キャッシュブロックサイズ）でメモリ内に整列された４ワードの命令のブロックに対するリクエストがバスインタフェースユニット（ＢＩＵ）２６０に対して行なわれる。これは、後続のミスが起こるということを仮定して、命令ブロックの継続するプリフェッチストリームを開始する。この特定の実施例では、キャッシュ内のブロックにつき有効ビットは１つしかないので、４ワードのブロックが最小の転送サイズである。有効ビットは、現在の１６バイトエントリおよびタグが有効であることを示す。このことは、エントリがロードされ、現在実行されているプログラムに対して確立されたことを意味する。

【００７２】命令ブロックが戻される際に（対象のワードからではなく下位のワードから行なわれる）、これは１つの命令につき４ビットの情報を発生するプリデコードネットワーク（図示せず）を通る。前の命令ブロックが発行されていれば、次の命令ブロック（新しい命令ブロック）が命令レジスタ２５８およびＩＤＥＣＯＤＥ２１０に進む。そうでなければ、次の命令ブロックはプリフェッチバッファ２５９で待つ。命令レジスタ２５８は、推論実行のために発行されるべき次の命令である現在の４つの命令を保持する。プリフェッチバッファ２５９は、ＩＣＡＣＨＥ２０５がリクエストしたプリフェッチされた命令のブロックを保持する。これらの命令は、後にプリデコードされてＩＣＡＣＨＥ２０５およびＩＤＥＣＯＤＥ２１０に送られる。この態様でプリフェッチされた命令のブロックを保持することによって、ＩＤＥＣＯＤＥ２１０による発行およびプリフェッチがロック状態で実行される必要がないように、バッファ動作が与えられる。

【００７３】まだ解決されていない条件付分岐がなければ、予測実行された次の命令がデコードに進むと、次の命令ブロックがＩＣＡＣＨＥ２０５に書込まれる。このアプローチは、望ましいことには不必要な命令がキャッシュされることを防ぐ。プリデコード情報もまたキャッシュに書込まれる。プリデコード情報とは、特定の命令を適切な機能ユニットに迅速に送るのを助ける命令のサイズおよび内容に関した情報である。プリデコードに関するさらなる情報は、同時係属中の本譲受人に譲渡された「可変バイト長命令に特に適したプリデコード命令キャッシュおよびそのための方法」（"Pre-Decoded Instruction Cache And Method Therefor Particularly Suitable For Variable Byte-Length Instructions"）と題される米国特許出願番号第１４５，９０５号に見いだされ、その開示をここに引用によって援用する。分岐予測は、プログラムが実行される際にどの分岐が発生するかを予測するために用いられるものであることが認められる。予測は後に、分岐が実際に実行されるときに確立される。予測は、マイクロプロセッサパイプラインのフェッチ段階の間に起こる。

【００７４】プリフェッチストリームは、ＢＩＵ２６０がそれに結合される外部バス（図示せず）を放棄しなくてはならないか、データキャッシュ２４５が外部アクセスを必要とするか、プリフェッチバッファ２５９がオーバーフローするか、キャッシュヒットが起こるか、または分岐もしくは割込が起こるまで続く。上述のことより、プリフェッチストリームはあまり長くはならない傾向にあることが認められるであろう。一般に、外部プリフェッチは、多くても発行されているものより２ブロック先である。

【００７５】この特定の実施例では、命令キャッシュ２

０５（ＩＣＡＣＨＥ）内のブロック１つにつき有効ビットは１つなので、部分的なブロックは存在せず、すべての外部フェッチは４つの命令のブロックで行なわれることが認められる。キャッシュ内のブロックにつき有効ビットは１つしかない。ＩＣＡＣＨＥ２０５はまた、各ブロックについての分岐予測情報を含む。この情報はリフィルの際にクリアされる。

【００７６】命令がＩＣＡＣＨＥ２０５に進んだので、スーパースカラ実行を始めることができる。外部でフェッチされたブロックがデコードに進むと、動作はＩＣＡＣＨＥ２０５からフェッチされたのと同じであるが、全体的な性能は、１サイクルにつき１の命令の最大外部フェッチレートに制限される。４ワードの命令ブロックがフェッチされ、プリデコード情報とともにデコードに進む（ＰＨ２でキャッシュ読出、ＰＨ１で命令バス駆動）。ＰＨ１はクロックの２つの相のうちの第１のものと規定され、ＰＨ２は、クロックの２つの相のうちの第２のものと規定される。ＰＨ１およびＰＨ２が、パイプライン化されるプロセッサの基本的なタイミングを構成する。

【００７７】図３および４に示されるように、３２ビットフェッチＰＣ（ＦＰＣ）バス、ＦＰＣ（３１：０）は、命令キャッシュ（ＩＣＡＣＨＥ）２０５とデコーダ（ＩＤＥＣＯＤＥ）２１０のフェッチャ２５７との間に結合される。より特定的には、ＦＰＣバスは、ＩＣＡＣＨＥ２０５内のＦＰＣブロック２０７とフェッチャ２５７との間に延びる。命令キャッシュ２０５内のフェッチＰＣまたはＦＰＣブロック２０７は、その中に位置されるＦＰＣとして示される推論フェッチプログラムカウンタを制御する。ＦＰＣブロック２０７は、デコーダ２１０による機能ユニットへの命令の発行に先立ってフェッチャ２５７がプリフェッチする命令に関連するプログラムカウント値ＦＰＣを保持する。ＦＰＣバスは、ＩＣＡＣＨＥに例外または分岐予測に進む位置を示す。フェッチＰＣブロック２０７は、デコーダ２１０へと命令（４の幅）をプリフェッチするのに、命令キャッシュ２０５にストアされた分岐予測情報を用いる。フェッチＰＣブロックは、逐次アクセスを予測することもでき、この場合には新しいブロックが必要なときに現在のフェッチＰＣを１６バイトだけ増分し、これはまた新しいブロックへの分岐を予測することもできる。新しい分岐位置は、予測された分岐に関して命令キャッシュから受取られたものでも、誤予測または例外の際に分岐機能ユニットから受取られたものでもあり得る。フェッチＰＣまたはＦＰＣは、先に述べたリタイヤＰＣとは区別されるべきである。

【００７８】フェッチＰＣ（ＦＰＣ）はＰＨ１で増分され、次ブロックがＩＣＡＣＨＥ２０５から読出されるが、ＩＤＥＣＯＤＥ２１０は、第１のブロックからすべての命令を発行していなければＨＯＬＤＩＦＥＴをアサートすることによってフェッチャ２５７を停止させる。ＨＯＬＤＩＦＥＴ信号の機能は、命令レジスタ２５８内の４つの命令が進むことができないので命令のフェッチを抑えるというものである。

【００７９】フェッチャ２５７はまた、分岐予測の実行を助ける。分岐予測は、命令キャッシュ２０５の出力である。分岐が予測されると、予測された次ブロックの４つの命令は、命令キャッシュ２０５によって命令ラインＩＮＳ０、ＩＮＳ１、ＩＮＳ２およびＩＮＳ３へと出力される。命令キャッシュ２０５内のアレイＩＣ＿ＮＸＴＢＬＫ（図示せず）は、キャッシュ内の各ブロックについてその特定のブロックでどの命令が予測実行されるかを規定し、次ブロックがどう予測されるかを示す。分岐がなければ、実行は常にブロック単位で逐次的に行なわれるであろう。したがって、発生される分岐は、このブロック指向分岐予測を変える唯一の事象である。言い換えれば、この発明の一実施例では、逐次的なブロック単位での予測は、発生しないと予測された分岐が発生し、誤予測されたときのみ起こる。

【００８０】分岐命令を含むブロックが初めてデコード２１０（ＩＤＥＣＯＤＥ）に送られると、後続のフェッチは、分岐が発生されないと仮定して、逐次的である。分岐が実行され、後に実際に発生したとわかると、分岐予測ユニット（分岐ユニット）５２０は、ＩＣＡＣＨＥ２０５に知らせ、１）分岐が発生したこと、２）分岐命令のブロック内の位置、および、３）ターゲット命令のキャッシュ内の位置を反映するように、そのブロックに関する予測情報を更新する。フェッチャ２５７はまた、ターゲットからフェッチを始めるように指示し直される。次にそのブロックがフェッチされると、フェッチャ２５７は、それが前に発生された分岐を含むことを認め、以下の動作で非逐次的フェッチを行なう、すなわち１）命令有効ビットは、分岐遅延スロットを含みかつそこまでしかセットされない。分岐遅延は常に分岐の後の命令を実行するという概念であり、遅延分岐とも称される。この命令は既にスカラＲＩＳＣパイプラインにおいてフリフェッチされており、そのため分岐の際に、それを実行するのにオーバーヘッドが失われない。２）分岐が発生予測されたという指示がそのブロックとともにデコーダ２１０に送られる。３）次のフェッチのためのキャッシュインデックスが予測情報からとられる。（キャッシュインデックスは、分岐が起こるときに予測実行された次ブロックのためのキャッシュ内の位置である。キャッシュインデックスは絶対ＰＣでないことに注目されたい。絶対ＰＣは、その位置のＴＡＧをキャッシュインデックスと連結することによって形成される。）４）このキャッシュインデックスのブロックがフェッチされ、予測されたターゲットアドレスがブロックのタグから形成され、分岐情報が分岐ＦＩＦＯ（ＢＲＮ　ＦＩＦＯ）２６１に置かれる。５）この次ブロックのための有効ビ

ットが、予測されたターゲット命令から始まってセットされる。

【００８１】分岐ＦＩＦＯ２６１は、フェッチャ２５７によって予測されたターゲットアドレスを分岐機能ユニット（ＢＲＮＳＥＣ）５５０に伝えるために用いられる。別個に示されているが、分岐ＦＩＦＯ２６１は分岐セクションＢＲＮＳＥＣ５５０の一部であると考えられることが認められる。分岐ＦＩＦＯ２６１には、ターゲットとともに分岐が発生予測された命令のＰＣがロードされる。分岐命令が実際に発行されると、分岐命令は分岐ＦＩＦＯ内のエントリ、すなわちそこにストアされたＰＣと比較される。一致があれば、エントリは分岐ＦＩＦＯから送られ、分岐命令がうまく予測されたものとしてリオーダバッファ２４０に戻される。誤予測があれば、正しいＰＣがリオーダバッファ２４０に与えられる。

【００８２】予測ビットは、分岐命令とともにデコーダ２１０によって分岐ユニット５２０に発行される。予測ビットは、特定の分岐がＩＣ＿ＮＸＴＢＬＫアレイにストアされた情報から発生予測されたかどうかを示す。

【００８３】分岐ユニット５２０が命令を実行すると、その結果が予測と比較され、発生されれば、実際のターゲットアドレスが分岐ＦＩＦＯの上部のエントリ（必要であればそれが現われるの待つ）と比較される。いずれのチェックも失敗すれば、分岐ユニット５２０はフェッチャ２５７に正しいターゲットアドレスを再指定し、予測を更新する。これがフェッチャ２５７によるものではなく予測された非順次的フェッチに関してキャッシュミスを検出する方法であることに注目されたい。予測情報は、フルアドレスではなくキャッシュインデックスのみを含むので、ターゲットブロックのタグはヒットに関してチェックすることができず、ターゲットアドレスはそのタグによって特定されるそのインデックスのブロックのアドレスであると仮定される。分岐が最後に実行されてから実際のターゲットブロックが置換えられていれば、これは誤比較および実行の際の訂正となる。誤比較が起これば、分岐を過ぎた多くの命令が、その遅延スロットのみだけでなく、実行されていいかもしれない。

【００８４】分岐予測ユニット５２０として用いることのできる分岐予測ユニットの１つは、１９９２年８月４日に発行された、ダブリュー・エム・ジョンソン（Ｗ．Ｍ．Johnson）の「キャッシュ内の各命令ブロックとストアされたフェッチ情報を用いた正しく予測された分岐命令に続く実行の遅延を減じるためのシステム」と題された米国特許番号第５，１３６，６９７号に説明され、その開示はここに引用によって援用される。

【００８５】［ＩＩＩ（ｂ）　命令フロー－デコード、レジスタファイル読出、発行］命令は１度に１ブロックずつＩＤＥＣＯＤＥ２１０に進み、それらのメモリブロック内の位置に対応する命令レジスタ２５８内の特定の位置を占める（０＝列の最初）。各命令に付随するのは、そのプリデコード情報および有効ビットである。

【００８６】ＩＤＥＣＯＤＥ２１０の主な機能は、命令を扱う機能ユニットに従って命令を分類し、その命令をそれらの機能ユニットに発行することである。これは、４つの３ビット命令タイプコード（ＩＮＳＴＹＰｎ）をすべての機能ユニットにブロードキャストし、何らかの所与のサイクル内で、発行されている各命令のための信号（ＸＩＮＳＤＩＳＰ（３：０））をアサートすることによって行なわれる。（本明細書中、Ｘ指示を伴って現われる信号と、伴わない信号とがある。ＸＩＮＳＤＩＳＰ信号等のＸは、誤ったアサートがバスを放電することを示す。）図３ないし図５に示されるように、マイクロプロセッサ５００は、タイプコードを機能ユニットにブロードキャストする目的のために４のタイプバス、ＩＮＳＴＹＰｎ（７：０）を含む。特定の命令ブロックの４つの命令の各々についてそれぞれのＴＹＰＥバスが設けられる。

【００８７】特定の機能ユニットがそのタイプに対応するＴＹＰＥ信号を検出すると、その機能ユニットは、タイプバスにおいて検出されたタイプ信号の位置に従って、ＩＤＥＣＯＤＥ２１０の現在の発行ウィンドウ内の現在の命令ブロックの４つの命令のうちのどれを受取るべきかを知る。タイプバスは、ＩＤＥＣＯＤＥ２１０のそれぞれの発行位置に対応する４つのセクションを有する。その機能ユニットはまた、検出されたタイプに対応する発行情報バスのそのセクションで起こる操作コード（ｏｐコード）によってその命令のオペランドデータにどの機能を実行するべきかを定める。さらに、機能ユニットはどの命令を実行すべきかがわかっているので、そのハードウェアをオペランドデータと行先タグとを受取るためのオペランドデータバスおよびそれぞれの行先タグバスＤＥＳＴ．ＴＡＧ（０：３）と整列させる。

【００８８】命令が発行されると、それらの有効ビットはリセットされ、そのタイプは「空」になる。特定のブロックの４つの命令すべてが、命令の次ブロックがフェッチされる前に発行されなくてはならない。ブロックの４つの命令すべてが１度に発行されてもよいが、以下の事象が起こる可能性があり、それもよく起こるので、このプロセスを遅くする。

１）　クラスの競合－これは２つまたはそれ以上の命令が同じ機能ユニットを必要とするときに起こる。整数コードはマイクロプロセッサ５００にとって重要である。この理由のため、本発明の一実施例は、機能ユニットＡＬＵ０、ＡＬＵ１、ＳＨＦＳＥＣ、ＢＲＮＳＥＣ、ＬＳＳＥＣ、ＦＰＴＳＥＣおよびＳＲＢＳＥＣの間でクラスの競合が起こるのを減じるために２つのＡＬＵを含む。命令は直列化の点でのみＳＲＢＳＥＣ５１２に発行される。言い換えれば、直列に実行されなくてはならない命令のみがＳＲＢＳＥＣ５１２に送られる。

2）　機能ユニットが命令を受取ることができない
3）　レジスタファイル（ＲＦ）２３５のポートが利用可能でない－この実施例において、８つのオペランドバスを与えるために通常考えるような８つではなく４つのＲＦ読出しポートしか存在しない。命令の多くはレジスタファイル２３５から２つのオペランドを必要とすることはなく、またはＲＯＢ２４０によるオペランド転送によって満たされ得るために、読出しポートの数がこのように少ないことは最初に考えるほどは制限的ではないことがわかった。たとえば８つの、より多くのＲＦ読出しポートを用いて、レジスタファイルポートが利用可能でない状態が起こる可能性を避けるような、この発明の他の実施例も企図される。
4）　リオーダバッファ２４０におけるスペースの欠如－各命令は対応するリオーダバッファのエントリを持たなくてはならず（または倍および拡張精度浮動小数点命令の場合のように、２つのリオーダバッファエントリが設けられる）、リオーダバッファはＲＯＢＳＴＡＴ（３：０）によって、予測された命令のうちのいくつに場所を見つけられるかを示す。図３および４に示されるように、ＲＯＢＳＴＡＴ（３：０）と示される状態バスが、リオーダバッファ（ＲＯＢ）２４０とデコーダ（ＩＤＥＣＯＤＥ）２１０との間に結合される。ＲＯＢＳＴＡＴ（３：０）は、ＲＯＢからＩＤＥＣＯＤＥに、４つの現在の命令のうちのいくつが割当てられるＲＯＢエントリを有するかを示す。ここでＲＯＢのエントリを充満することが可能であることに注目されたい。
5）　直列化－命令の中には逐次状態を守る機構の範囲を越えた状態を変更するものがある－これらの命令（たとえばＭＴＳＲ、ＭＦＳＲ、ＩＲＥＴ命令）は周りの命令に関してプログラム順に実行されなくてはならない。

【００８９】上に挙げた５つの状況のうちの１つが起これば、影響を受ける命令は発行を停止し、後続の命令は、それらを抑えるものが他に何もなくても発行され得ない。各発行位置について、機能ユニットにソースオペランドを供給するＡおよびＢオペランドバスの組（ＸＲＤｎＡＢ／ＸＲＤｎＢＢバスとも称される）がある。レジスタファイル２３５はデコードと並列にＰＨ２でアクセスされ、オペランドがＰＨ１でこれらのバスに送られる。ソースレジスタを変更する命令がまだ実行中であれば、レジスタファイル２３５内の値は無効である。このことは、レジスタファイル２３５およびＲＯＢ２４０がデータを含まず、したがってタグがデータの代わりとなることを意味する。リオーダバッファ（ＲＯＢ）２４０はこれを追跡し、レジスタファイルアクセスと並列してアクセスされる。オペランドが利用可能でないこと、またはレジスタの競合は発行の際に問題とならないことに注目されたい。ＲＯＢ２４０は、予め定められた数のエントリならびに先頭および末尾ポインタを備えた循状バッファとして見なすことができる。

【００９０】命令が発行されると、ＲＯＢ内のエントリがその行先レジスタのために確保される。ＲＯＢ内の各エントリは、１）命令の行先レジスタアドレス、２）命令の結果のためのスペース（これは倍精度動作またはＣＡＬＬ／ＪＭＰＦＤＥＣタイプの命令には２つのエントリを必要とするかもしれない）、および例外状態情報および、３）ａ）エントリが割当てられたことと、ｂ）結果が戻されたこととを示すビットからなる。

【００９１】エントリは末尾ポインタから始まって逐次的に割当てられる。割当ビットは、セットされて命令が発行されたことを示す。割当ビットは各ＲＯＢエントリと関連付けられる。割当ビットは、特定のＲＯＢエントリが未処理の動作に割当てられたことを示す。割当ビットは、エントリが用済となると、または例外が起こると割当から外される。別個の有効ビットが、結果が完了されレジスタファイルに書込まれたかどうかを示す。エントリのアドレス（結果または行先タグとも呼ばれる）が発行から実行の間対応する命令に付随し、結果バスの１つを介して命令の結果とともにＲＯＢ２４０に戻される。

【００９２】より詳細には、行先タグは、命令が機能ユニットに発行されるときに用いられ、結果タグは命令が戻されるとき、すなわち結果が機能ユニットからＲＯＢに戻されるときに用いられる。言い換えれば、行先タグは発行された命令に関連し、リオーダバッファによって機能ユニットに特定の命令の結果がどこにストアされるべきかに関して知らせるために機能ユニットに与えられる。

【００９３】より詳細には、命令に関連する行先タグは機能ユニットにストアされ、次に結果バスに転送される。このような行先タグは、これらが結果バスを介して転送されるときにはまだ行先タグとして示される。これらのタグは他の機能ユニットの待合わせステーションでオペランドタグと比較され、このような他の機能ユニットが特定の結果を必要かどうかを見る。特定の機能ユニットからの結果は、ＲＯＢ内の対応する相対推論位置に戻される。

【００９４】命令の結果は、効果的にこの命令の結果タグとなる命令の行先タグによって識別されるＲＯＢエントリ内に置かれる。その特定のＲＯＢエントリの有効ビットがセットされる。結果は、レジスタファイルにライトバックされる順番が囲ってくるまでそこに留まる。エントリが除去されるよりも早くＲＯＢ２４０に割当てられることが可能であり、この場合にはＲＯＢ２４０は最終的にはフルとなる。リオーダバッファフル状態は、ＲＯＢＳＴＡＴ（３：０）バスを介してデコーダ２１０に伝えられる。これに応答して、デコーダ２１０はＨＯＬＤＩＦＥＴ信号を発生して、命令がＩＣＡＣＨＥ２０５からフェッチされるのを止める。したがって、ＲＯＢフル状態はデコーダ２１０による発行を止めることが認め

られる。

【0095】オペランドの処理の説明に戻って、ROB240でライトバックを待っている結果を、もし必要であれば他の機能ユニットに転送することができることに注目されたい。これは、IDECODE210内の命令のソースレジスタアドレスをROB内の行先レジスタアドレスと、デコード時にレジスタファイルアクセスと並列して、比較することによって行なわれる。AおよびBソースオペランドに関して起こり、かつ結果有効ビットがセットされている、最も最近のアドレス一致について、ROB240は対応する結果をレジスタファイル235の代わりに適切なオペランドバスに送る。この一致が起これば、ROB240は、ROB240とレジスタファイル235との間のOVERRIDEラインを活性化して、レジスタファイル235に、AおよびBオペランドバスにいかなるオペランドも送らないように指示する。

【0096】たとえば、デコーダ210が、レジスタR3の内容をレジスタR5の内容に加えてその結果をレジスタR7に置くことを意味するように規定される、命令ADD　R3、R5、R7をデコードしていると仮定する。この例において、IDECODE内でデコードされるソースレジスタアドレスR3およびR5は、ROB240内の行先レジスタアドレスと比較される。この例の目的のため、結果R3がROB240内に含まれ、結果R5がレジスタファイル235内に含まれると仮定する。これらの状況のもとでは、デコードされた命令内のソースアドレスR3とROB240内の行先レジスタアドレスR3との比較は肯定である。レジスタR3のためのROBエントリの結果がROB240から検索され、適切な機能ユニット、すなわちALU0またはALU1の待合わせステーションによるラッチのためにオペランドAバスにブロードキャストされる。この場合にROBエントリと一致が見いだされるので、レジスタファイル235が、それが含み得る何らかの用済となったR3値でAオペランドバスを駆動しないように、OVERRIDEラインが駆動される。

【0097】この例で、デコードされた命令内のソースアドレスR5とROB240内に含まれる行先レジスタアドレスとの比較はうまく行かない。したがって、レジスタファイル235内に含まれる結果値R5がBオペランドバスへ駆動され、その結果が機能ユニットすなわちALU0に実行のためにブロードキャストされる。Aオペランドおよびオペランドの両方がALU0機能ユニットの待合わせステーション内にあれば、命令がALU0に投入されて、ALU0によって実行される。結果（結果オペランド）は、この結果オペランドを求めている他の機能ユニットの待合わせステーションに送るために結果バス265に置かれる。結果オペランドはまた、その結果のために割当てられたエントリでそこにストアするためにROB240にも与えられる。

【0098】所望のオペランド値がまだROB240になくても（アサートされる有効ビットによって示される）、それでも命令をデコーダ210によって発行することができる。この場合に、ROB240は一致するエントリのインデックス（すなわちその結果を最終的に生成する命令の結果タグ）を機能ユニットにオペランドの代わりに送る。ここでもやはり、8つのオペランドバスに対応する効果的に8つのA／Bタグバス（すなわち4つのAタグバスおよび4つのBタグバス、すなわちTAGnAB（4：0）およびTAGnBB（4：0）ここでnは整数である）があることに注目されたい。タグの最上位ビット（MSB）は、タグが有効であるときを示す。

【0099】2つ以上のROBエントリが同じ行先レジスタタグを有するときには、最も最近のエントリが用いられる。これは、可能である並列性を減じてしまうであろう独立した命令による行先としての同じレジスタの異なる使用を区別する。（これはライトアフターライトハザードとして知られる）命令のキャッシュ化の際に発生されるプリデコード情報はデコード時に作用し始める。プリデコード情報は、ICACHE205からPREDECODEラインを介してIDECODE210に渡されることが認められる。

【0100】プリデコードは以下の態様で行われる。各命令について、ROBエントリの割当を、いくつのエントリが必要であるかを示すことによって（エントリを1つ必要とする命令もあるし、2つのエントリを必要とする命令もある）遮める2ビットコードを含むプリデコード信号PREDECODEがある。たとえば、加算命令ADD　（RA＋RB）→RCは、レジスタRC内に置かれるべき単一の32ビット結果のために1つのエントリを必要とする。対照的に、乗算命令DFMULT（RA＋RB）（倍精度）は、64ビットの結果を保持するのに2つのROBエントリを必要とする。本発明の特定の実施例では、各ROBエントリは32ビット幅である。この2ビットコードはさらに、所与の命令かもいくつの結果オペランドが生じるかを示す（すなわち、なし－分岐等、1－ほとんどのもの、または2－倍精度）。プリデコード情報は、レジスタファイルアクセスがAおよびBオペランドに必要であるかどうかを示す2つの付加的なビットを含む。したがって、マイクロプロセッサ500において32ビット命令につき4ビットのプリデコード情報がある。これらのビットはPH2のアクセスに先立って、PH1でレジスタファイルポートの効率的な割当を可能にする。命令が必要とするレジスタファイルポートを割当てられていないが、ROB240がオペランドを転送できることを示していれば、いずれにしても命令は発行され得る。

【0101】[III（c）　命令フロー－機能ユニッ

ト、待合わせステーション］図3ないし図5は、マイクロプロセッサ500のすべての機能ユニットが共通のデータ処理バス535上にあることを示す。データ処理バス535は、その比較的広いバンド幅のために高速のバスである。各機能ユニットにはその入力で2つの待合わせステーションが備えられている。より多いまたは少ない待合わせステーションが機能ユニットで用いられる本発明の他の実施例も企図される。

【0102】整数ユニット515は算術論理装置ALU0およびALU1を含む。ALU0には待合わせステーション540が設けられ、ALU1には待合わせステーション545が設けられる。分岐ユニット520（BRNSEC）にはその入力で待合わせステーション550が供給される。浮動小数点ユニット（FPTSEC）525は、浮動小数点加算ユニット555を含み、これには待合わせステーション560が設けられる。浮動小数点ユニット525はさらに、浮動小数点変換ユニット565を含み、これには待合わせステーション570が設けられる。浮動小数点ユニット525はさらに、浮動小数点乗算ユニット575を含み、これには待合わせステーション580が備えられる。最後に、浮動小数点ユニット525はさらに、浮動小数点除算ユニット585を含み、これにはその入力で待合わせステーション590が備えられる。ロード／ストアユニット530もまた、データ処理バス535上に存在し、待合わせステーション600を含む。

【0103】図3ないし図5に示されるように、各機能ユニットへの主入力（すなわち機能ユニットと関連する各待合わせステーションへの入力）は、以下の主データ処理バス535を構成するバスによって与えられる、すなわち

1）　IDECODE210からの4つのOPCODEバス（INSOPn（7：0）として示され、nは0ないし3の整数である）

2）　IDECODE210からの4つの命令タイプバス（INSTYPn（7：0）として示され、nは0ないし3の整数である）

3）　IDECODE210からの4つの4ビット発行ベクトルバス（XINSDISP（3：0）として示される）

4）　Aオペランドバスおよびオペランドバスの4つの対（XRDnAB／XRDnBB（31：0）と示され、nは0ないし3の整数である）

5）　関連するA／Bタグバスの4つの対（TAGnAB／TAGnBB（4：0）と示され、nは0ないし3の整数である）

6）　3つの双方向結果オペランドバスを含む結果バス265（XRES0B（31：0）、XRES1B（31：0）、XRES2B（31：0）として示される）

7）　2つの結果タグバス（XRESTAG0B／SRESTAG1B（2：0）として示される）および

8）　2つの結果状態バス（XRESSTAT0BおよびXRESSTAT1B（2：0）と示される）である。

【0104】1つ以上の待合わせステーションが上述の機能ユニットの各々の前部に置かれる。待合わせステーションは、本質的には、機能ユニットによる実行を待ちながらそこで命令が待ち行列にされる先入れ先出し（FIFO）バッファである。命令がオペランドの代わりにタグを伴って発行されれば、または機能ユニットが停止またはビジー状態であれば、命令は待合わせステーションで待ち行列にされ、後続の命令はその後で待ち行列にされる（特定の機能ユニット内の投入は全くの順番通りであることに注目されたい）。待合わせステーションが充満すれば、これを示す信号がIDECODEにアサートされる。これは、同じタイプの別の命令に出会えば、発行を止める。

【0105】命令の発行は以下のように起こる。各待合わせステーションは対応する命令タイプに関して命令TYPEバスを（PH2で）観察する待合わせステーション論理を含む。待合わせステーションは、対応するopコード、AおよびBオペランドならびにAおよびBオペランドタグバスを、このような命令タイプに出会えば選択する。関連する機能ユニットで実行する2つ以上の命令が認められれば、プログラム順に関して先の命令が優先される。しかしながら、対応する発行ビットがセットされていることを認めるまで（PH1でXINSDISP（n））、命令は待合わせステーションに受入れられない。

【0106】この時点で、必要とされるオペランドが利用可能であり、かつ機能ユニットが何らかの理由のために停止されているわけでも、またはビジーであるわけでもなく、さらに前の命令が待合わせステーションで待っていなければ、命令は直ちに同じクロックサイクル内で実行に移る。そうでなければ、命令は待合わせステーションに置かれる。命令がオペランドの代わりにオペランドタグを、伴って発行されていれば、待合わせステーション論理は、オペランドタグを結果タグバス（XRESTAG0BおよびXRESTAG1B）で現われる結果タグと比較する。一致が認められれば、その結果が結果バス群265の対応する結果バスから取入れられる。この結果は次に、命令を投入するのを可能にすれば機能ユニットに転送される。そうでなければ、結果はオペランドとして待合わせステーションに置かれ、ここで命令を完了するのを助け、対応するタグ有効ビットはクリアされる。両方のオペランドが、汎用結果バスのいずれかまたは両方から同時に転送され得ることに注目されたい。

【0107】結果バス265を形成する3つの結果バスは、2つの汎用結果バスXRES0B（31：0）およびXRES1B（31：0）を含み、さらに分岐および

ストア専用の1つの結果バスXRES2B（31：0）を含む。結果バスXRES2B（31：0）は分岐およびストア専用なので、これが処理する結果（たとえば分岐PCアドレス等）は転送されない。機能ユニットは結果バスXRES0B（31：0）およびXRES1B（31：0）をモニタし、一方リオーダバッファ（RB）240は3つの結果バスすべてをモニタする。

【0108】命令が待合わせステーションで待つ際に、何らかの有効オペランドタグも同様に結果タグと比較され、同じような転送が行なわれる。機能ユニット間および機能ユニット内での結果の転送がこの態様で行なわれる。待合わせステーションと関連して、このタグの付与によって、従属性の適切なシーケンシングを維持しながら、異なる機能ユニットで順序通りでない命令の実行を可能にし、さらにオペランドハザードが無関係の後続の命令の実行をブロックすることを防ぐ。命令タイプおよびA／BタグはPH2で利用可能であり、一方投入する決定は後続のPH1で行なわれる。

【0109】待合わせステーションのオペランドは、これらが送られた実際のオペランドデータでなければ、タグおよび有効ビットを有する。言い換えれば、命令が待合わせステーションに発行され、かつ特定のオペランドがまだ利用可能でなければ、そのオペランドに関連するオペランドタグが実際のオペランドの代わりに待合わせステーションに与えられる。有効ビットは各オペランドタグと関連する。結果が機能ユニットで完了すると、結果は他の機能ユニットおよびROB240に結合される結果バスに与えられる。結果は待合わせステーションのオペランドタグと比較されて、ヒットが起これば、タグ有効ビットがクリアされて、結果バスからのオペランドは、オペランドに対して指定された機能ユニットの位置に転送される。言い換えれば、待合わせステーション内の何らかのエントリに一致する結果タグ0および1における タグ比較が値をそのステーションに転送する。

【0110】どの命令源（待合わせステーションまたは待合わせステーションに結合される4つの入来するバスのうちの1つ）が局所的デコードの次の候補であるかを定め、待合わせステーションの先頭にあるエントリに関する待合わせステーション有効ビットおよびデコード／優先命令タイプバスを調べることによってPH2で投入が行なわれ、この際に待合わせステーションのエントリが優先する。待合わせステーションを2つ有する機能ユニットでは、その2つの待合わせステーションは先入れ先出し（FIFO）構成を形成し、待合わせステーションに発行される第1の命令がFIFOの先頭を形成し、FIFOに発行される最後の命令がFIFOの末尾を形成する。

【0111】機能ユニットによる局所的デコードとは、タイプバスをモニタすることによって、機能ユニットがまず、そのタイプの命令が発行されていることを定めるということを意味する。一旦機能ユニットが、それが処理すべき命令を識別すると、機能ユニットはopコードバス上の対応するopコードを調べて、機能ユニットが実行すべき精確な命令を判断する。

【0112】本発明のこの実施例では、実行時間は、特定の命令タイプおよびその命令を実行する機能ユニットに依存する。より具体的には、すべてのALU、シフタ、分岐動作およびキャッシュでヒットするロード／ストアの1サイクルから、浮動小数点、ロード／ストアミスおよび特殊レジスタ動作のための数サイクルにまでわたる。特殊レジスタとは、再指定されない何らかの汎用でないレジスタと規定される。

【0113】機能ユニットは以下のように結果バスに対して調停する。結果バス2は、オペランドを戻さないストアのため、および計算されたターゲットアドレスを戻す分岐のために用いられる。分岐には優先順位があることが認められる。汎用結果バス0および1は、ALU0またはALU1のいずれかから、シフタユニット510から、浮動小数点ユニット525からの結果とロードおよび特殊レジスタアクセスとを扱う。

【0114】結果バス0（XRES0B（31：0）とも示される）および結果バス1（XRES1B（31：0）とも示される）へのアクセスを得ることに関する機能ユニット間での優先順位は、図6に示される。図6の表において、「DPの下位半分」という用語は、倍精度数の下位半分を意味する。マイクロプロセッサ500は、倍精度（DP）数を送るのに32ビットオペランドバスを用いる。より具体的には、倍精度数がオペランドバスを介して伝送されるとき、その数は2つの32ビット部分、すなわち上位32ビット部分と下位32ビット部分とで伝送される。上位および下位部分は、一般に2サイクルで2オペランドバスを介して伝送される。機能ユニットによる特定の結果バスに対するアクセスのリクエストの拒否は、その機能ユニットを停止させ、待合わせステーションフル状態としてデコードにさせるためにに戻り得る。

【0115】結果は、結果のタイプ（なし、通常または例外、および命令固有のコード、すなわちデータキャッシュミス、アサートトラップおよび分岐誤予測）を示す3ビット状態コード（RESULT STATUS）を含む。一実施例では、結果はまた、そのユニットおよび命令に依存して、32ビット結果オペランドおよび詳細な実行または例外状態を含む。結果バス235は、結果をROB240に戻すため、および結果を機能ユニットの待合わせステーションに転送するために用いられる。結果情報のすべてがROB240にストアされるが、機能ユニットは結果状態コードおよび結果オペランドを見るだけである。

【0116】ほとんどの機能ユニットは上述の態様で動作する。しかしながら、特殊レジスタブロックセクショ

ン（ＳＲＢＳＥＣ）５１２およびロード／ストアセクション（ＬＳＳＥＣ）５３０は、いくぶん異なる。ＳＲＢＳＥＣ機能ユニットは、頻繁には更新されずかつレジスタ再指定によってサポートされない状態および制御レジスタ等のマシン状態情報を保持する。ＳＲＢＳＥＣ５１２の特殊レジスタへの、およびそこからの動きは、周りの命令に関して常に直列化される。したがって、ＳＲＢＳＥＣは、別個の機能ユニットでありながら、直列化のためにオペランドが常にレジスタファイル２３５から利用可能であるので、待合わせステーションを必要としない。ＳＲＢＳＥＣ機能ユニットによって実行される命令の例には、「スペシャルレジスタへ移動」ＭＴＳＲ、および「スペシャルレジスタから移動」ＭＦＳＲ命令がある。直列化を必要とするこのような命令を実行する前に、マイクロプロセッサ５００は、この命令の前のすべての推論状態を直列化するか、または実行する。アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドによって製造されるＡＭ２９０００マイクロプロセッサで用いられるのと同じ特殊レジスタブロックを、ＳＲＢＳＥＣ５１２として用いてもよい。

【０１１７】ロード／ストアセクションＬＳＳＥＣ５３０は、他の機能ユニットと同じ態様で待合わせステーションを用いる。ロード／ストアセクション５３０は、データキャッシュ２４５からのデータのロードおよびデータキャッシュ２４５におけるデータのストアを制御する。しかしながら、命令の実行に関して、これは最も複雑な機能ユニットである。ＬＳＳＥＣは、データキャッシュ（ＤＣＡＣＨＥ）２４５およびメモリ管理ユニット（ＭＭＵ）２４７と密に結合する。マイクロプロセッサ５００は、データキャッシュ２４５または主メモリ２５５を変更する何らかの動作が未完了となり得ないように設計される。さらに、このような変更は、周りの命令に関してプログラム順に起こらなくてはならない。このことは、すべてのストアおよびデータキャッシュでミスしているロードの実行がＲＯＢ２４０内のリタイア論理２４２と協働しなくてはならないことを意味する。このことは、対応するＲＯＢエントリにＲＯＢリタイア論理が出会うまでこれらの動作が待ち行列にされるＦＩＦＯである、アクセスバッファ６０５と呼ばれる機構を用いて行なわれる。

【０１１８】データキャッシュ（ＤＣＡＣＨＥ）２４５として用いることができるデータキャッシュの１つ、およびロード／ストアセクション（ＬＳＳＥＣ）５３０として用いることができる１つのロード／ストアセクションは、同時係属中であり本譲受人に譲受された「高性能ロード／ストア機能ユニットおよびデータキャッシュ」（“High Performance Load/Store Functional Unit And Data Cache”）と題される米国特許出願連続番号第１４６，３７６号に記載され、その開示はここに引用によって援用される。命令キャッシュおよびデータキャッシュのアドレス指定に関するさらなる情報は、同時係属中であり、本譲受人に譲受された「線形アドレス可能なマイクロプロセッサキャッシュ」（“Linearly Addressable Microprocessor Cache”）と題される同時係属中の米国特許出願連続番号第１４６，３８１号に記載され、その開示はここに引用によって援用される。

【０１１９】アクセスバッファ６０５はＬＳＳＥＣ５３０内に位置される。一実施例において、アクセスバッファ６０５はミスしているロードまたはストア（ヒット／ミス）の２－４ワードＦＩＦＯである。ヒットしているストアは、それが実行されるべき次のものとなるまで書込まれない。しかしながら、アクセスまたはストアバッファによって、この状態は一時記憶装置に保持されることが可能となり、これはＲＯＢがレジスタ参照を転送するのと類似した態様でデータ参照を転送することができる。アクセスバッファは最後に、アクセスバッファの内容がプログラム順で次であるときにデータキャッシュ２４５（ＣＡＣＨＥ）に書込む。言い換えれば、アクセスバッファまたはストアバッファは、他のロード／ストア命令が処理され続けることが可能であるように１つまたはそれ以上のロード／ストア命令をストアするＦＩＦＯバッファである。たとえば、アクセスバッファ６０５は、後続のロードがロード／ストアユニットＬＳＳＥＣ５３０によって実行されている一方で、ストアを保持することができる。

【０１２０】ストアバッファとしても知られるアクセスバッファ、およびデータキャッシュと関連して用いられるロード／ストア機能ユニットは、同時継続中で本譲受人に譲受された「高性能ロード／ストア機能ユニットおよびデータキャッシュ」と題される同時係属中の特許出願により詳細に述べられ、その開示をここに引用によって援用する。

【０１２１】ＲＯＢリタイア論理２４２の機能は、どの命令がＲＯＢ２４０からレジスタファイル２３５へと格納されるべきであるかを定めることである。ＲＯＢエントリのこの格納の基準は、エントリが有効かつ割当てられること、結果が機能ユニットから戻されていること、およびエントリが誤予測または例外事象でマークされていないことである。

【０１２２】ストア動作は２つのオペランド、すなわちメモリアドレスおよびデータを必要とする。ストアが投入されると、これはＬＳＳＥＣ待合わせステーション６００からアクセスバッファ６０５へと転送され、ストア結果状態がＲＯＢ２４０に戻される。ストアは、データがまだ利用可能でなくても投入され得るが、アドレスはそこになくてはならない。この場合、アクセスバッファは待合わせステーションと類似した態様でタグを用いて、結果バス２３５からストアデータを選択する。ストアが投入される際、メモリ管理ユニット（ＭＭＵ）２４７で高速変換バッファ（ＴＬＢ）６１５のルックアップ

が行なわれ、データキャッシュがアクセスされてヒットについてチェックする。

【0123】MMUからの物理アドレスおよび仮想アドレスのページ部分は、データキャッシュからのステータス情報とともにアクセスバッファに置かれる。言い換えれば、キャッシュは物理的にアドレスされる。TLBミスが起こると、これは結果状態に反映され、適切なトラップベクトルが結果バス2に送られ、この時点では他の動作は行なわれない。（ロードに関するTLBルックアップも同じように行なわれるが、何らかのトラップベクトルは結果バス1に進む。）トラップベクトルは例外である。マイクロプロセッサ500はTLBトラップを取込み、新しいページを物理メモリにロードして、TLBを更新する。この動作には数百サイクルかかる可能性があるが、比較的頻繁には起こらない事象である。マイクロプロセッサ500はPCを止めて、マイクロプロセッサレジスタをストアし戻し、ベクトルを実行して、レジスタ状態を復元し、割込リターンを実行する。

【0124】ストアがアクセスバッファの先頭に達すると（次いで空であればすぐに行なわれる）、ROB240が、対応するROBエントリが用済の段階に達したことを示すLSRETIREと符号を付される信号をアサートし、次いでキャッシュアクセスを進める。しかしながら、キャッシュが前のリフィルを完了させること、またはコヒーレンシー動作を行なうことでビジー状態であれば、遅延され得る。一方、ROB240は動作を続け、別のストア命令に出会うかもしれない。LSSECがそれを完了する準備ができる前にそのストア命令が用済とされないようにするために、以下のようにハンドシェイクが用いられる。LSSEC530はROB240に、LSDONEをアサートすることによってLSSECが動作を完了したときを示す信号を与える。ROB240は、前のストアが用済とされてからLSDONEを認めていなければ、ストア（またはロード）を停止することが認められる。

【0125】データキャッシュ245においてヒットしているロード動作は、ROB240と協働されなくてもよい。しかしながら、ミスはROB240と協働されて、不必要なリフィルおよび誤予測された分岐を越えての無効な外部参照を避けなくてはならない。ロードが投入されると、（キャッシュがビジー状態でなければ）キャッシュアクセスがすぐに行なわれる。キャッシュにおいてヒットがあれば、結果が通常状態コードとともに結果バスを介してROBに戻される。ミスがあれば、ロードはアクセスバッファ605に置かれ、ロード_ミス結果コードが戻される。ROB240のリタイア論理242がこの条件に出会えば、これはLSRETIREをアサートして、ロード_有効結果状態コードとともに結果バスに置かれている所望のワードから、これが現われるとすぐにリフィルが始まる（リフィルが終了するのを待たない）。ROB240は、ストアの場合のようにLSRETIREをアサートする際にロードを用済とできないことが認められる。その代わりに、ROB240はデータが戻るのを待たなくてはならない。

【0126】ロードは、アクセスバッファにおいて待っている、前の未完了のストア動作があっても処理され得る。ストアに関して順序通りでなくロードを行なうのを可能にする際に、マイクロプロセッサ500はロードが（プログラム順に関して）前のストアによってこれから変更される位置からは行なわれないことを確実にする。このことは、ロードアドレスをアクセスバッファ605内の何らかのストアアドレスと、キャッシュアクセスと並列して、比較することによって行なわれる。どれも一致しなければ、ロードは進められる。1つ一致するものがあれば（2つ以上の場合は最も最近のエントリ）、ストアデータがアクセスバッファ605からキャッシュデータの代わりに結果バス265に転送される。起こっているかもしれない何らかのキャッシュミスは無視される（すなわちリフィルは起こらない）。ストアデータがまだ存在しなければ、ロードはストアデータが到着するまで停止される。さらに、これらの動作は、望ましいことにはメモリアクセスが不必要に並列性を損なうことを防ぐ。

【0127】ここでさらにロード／ストアについて検討する。1Kバイトおよび2Kバイトページサイズに関して、高速変換バッファ（TLB）のルックアップが、キャッシュアクセスに先立って行なわれる。これはさらなるサイクルのロード／ストア待ち時間を起こす。LSSECがロードまたはストアを「完了する」とき、これは関連するキャッシュ動作が完了することを意味しないことに注目されたい。そうではなく、ICACHEまたはDCACHE、BIU、および外部でリフィル等の動作がまだあるかもしれない。

【0128】アクセスバッファ転送は、部分ワードロード／ストア動作のためには行なわれない。ワードアドレス一致が検出され、かつロードとストアとの間で何らかのオーバーラップがあれば、ロードはキャッシュミスのように見えるようにされ、ストアの後に実行されるようにアクセスバッファ605で待ち行列にされる（実際にはキャッシュでヒットしているかもしれないし、していないかもしれない）。オーバーラップがなければ、ロードはアドレス一致がなかったかのように進められる。

【0129】ロード／ストアマルチ命令は、直列化の態様で行なわれる、すなわちロード／ストアマルチ命令が実行されているとき、他のどの命令も並列して行なわれないことが認められる。ロードまたはストア（ロード／ストア）マルチ命令は、レジスタファイルへの、またはそこからのブロックの動きである。この命令は、所与のアドレス、所与のレジスタ、およびカウントフィールドを含む。ロード／ストアのマルチ命令の一例に、LOA

ＤＭ（Ｃ，Ａ，Ｂ）があり、Ｃは行先レジスタ、Ａはアドレスレジスタ、およびＢは転送の数である。

【０１３０】ロードミスは必ずしもリフィルを起こさないことも認められる。その代わりに、ページはキャッシュ不可能としてマークされるかもしれず、ロードがアクセスバッファから消されているかもしれない。

【０１３１】［ＩＩＩ（Ｄ）　命令フローリオーダバッファおよび命令リタイア］結果がＲＯＢ２４０に戻されると、これらは結果タグによって特定されるエントリに書込まれ、これはＲＯＢの先頭および末尾ポインタの間の何らかの場所にある。ライトバック、ストアおよびロードミスの実行、トラップおよびＰＣ０、ＰＣ１およびＰＣ２の更新を制御するリタイア論理２４２は、プログラム順に有効結果を伴うエントリを見る。

【０１３２】ＰＣ０、ＰＣ１およびＰＣ２は、ＤＥＣ、ＥＸＥＣおよびＷＲＩＴＥＢＡＣＫ０，１の値を含むマッピングレジスタである。信号ＤＥＣ、ＥＸＥＣおよびＷＲＩＴＥＢＡＣＫ０，１は、スカラＡＭ２９０００パイプラインからの段階であるデコード、実行およびライトバックを指し、ＡＭＤ２９００は、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドから入手可能なマイクロプロセッサである。これらの信号は、実行の際にパイプラインを再始動させるのに用いられる。遅延分岐のために２つ以上のＰＣが用いられる。ＰＣ０、ＰＣ１およびＰＣ２は、割込またはトラップの際に用いられて、分岐誤予測または例外に出会うとマイクロプロセッサ５００が戻り得る、ＤＥＣ、ＥＸＥＣおよびＷＲＩＴＥＢＡＣＫ０、１の古い値を保持する。ＰＣ０、ＰＣ１およびＰＣ２は、パイプラインを再始動させるために割込リターンの際に用いられ、リオーダバッファ２４０内のリタイア論理２４２内に含まれる。ＰＣ１は現在のリタイアＰＣをマッピングする。

【０１３３】通常の結果を有するエントリに出会えば、結果オペランド（もしあれば）がエントリにおいて特定されたレジスタファイル（ＲＦ）２３５の位置に書込まれる。ＲＦ書込ポート（ＷＲ）は２つあるので、２つのオペランドが同時にレジスタファイルに格納され得る。ＲＯＢ２４０は、さらに１つのストアおよび１つの分岐を用済とすることができ、最大で４つの命令を１マイクロプロセッササイクルについて用済とできる。

【０１３４】ＣＰＳビットおよびＦＰＳスティッキービット等の他の状態は、この時点で更新され得る。ＣＰＳは現在のプロセッサ状態を指し、ＣＰＳはプログラム状態およが条件コードレジスタを示す。ＦＰＳは浮動小数点状態レジスタビットを指す。ＦＰＳは、浮動小数点機能ユニット５２５のための状態／条件コードレジスタを示す。ＦＰＳスティッキービットとは、セット条件によってセットされ、クリア条件でクリアされないビットのことである。ＦＰＳスティッキービットは、浮動小数点数の丸め制御のために用いられる。たとえば、マイクロプロセッサ５００が値を減算するか、またはシフトすれば、いくつかの最下位ビット（ＬＳＢ）が仮数部からシフトされる。ＦＰＳスティッキービットは、この条件が起こったという指示を与える。

【０１３５】その結果がまだ戻されていないＲＯＢ２４０内のエントリは、結果が戻ってくるまでそれ以上の処理を停止させる。そのエントリを越えるものは、たとえ有効であっても用済とはされない。ストア結果に出会えば、ＲＯＢ２４０は、実際にストアを行なって命令を用済とするようにロード／ストアセクションにゴーアヘッド指示を与える。ロードミス結果に出会えば、ＲＯＢ２４０はロードを実行するようにゴーアヘッド指示を与える。ロードが完了すると、要求されたロードオペランドはＲＯＢ２４０にロードヒット状態とともに戻され、これが命令を用済とすることを可能にし、そのオペランドを待っている何らかの待合わせステーションによって認められる。分岐結果に出会えば、ＲＯＢ２４０はこれを用いてＰＣ１を更新する。

【０１３６】マイクロプロセッサのアーキテクチャ状態は、プログラム内のリタイアＰＣの現在の状態である。マイクロプロセッサの推論状態は、ＦＥＴＣＨＰＣの現在の値、デコーダおよびリオーダバッファ内のエントリのすべてである。これらは、ダイナミックに更新される現在の命令の推論キューである。例外または誤予測の際に、すべての推論状態はクリアされ得るが、アーキテクチャ状態は、これがレジスタファイルの現在の状態なので、クリアされ得ない。

【０１３７】誤予測分岐遅延スロットを越える命令は、誤予測が明らかとなる前に実行され得ることを先に述べた。この発生は、ＲＯＢ２４０によって区別される。誤予測が検出されると、いかなる未発行の命令もクリアされ、フェッチャ２５７が再び指示される。どの機能ユニットも誤予測を知らされない（しかしながら分岐ユニット５２０はその待合わせステーション５５０内の何らかの有効エントリにおける「キャンセル」ビットをセットし、そのためこれらの分岐は皆を受けずに実行され、誤予測を起こすことなくＲＯＢ２４０に戻される）。

【０１３８】このような誤予測が起こると、ＲＯＢ内の対応するエントリは誤予測されたものとして割当てられる。後続のエントリが機能ユニットから転送されるとき、これらは完了されているが誤予測されたものとしてマークされる。リオーダバッファ２４０内のリタイア論理２４２は、これらのエントリを無視して、割当から外す。

【０１３９】同時に、発生／非発生および正確／不正確な予測を示す分岐結果状態がＲＯＢ２４０に戻される。誤予測の結果は、ＲＯＢに、分岐エントリの後の２つ目から（遅延スロットを考慮して）末尾ポインタまでのすべてのエントリのキャンセルビットを直ちにセットさせる。この発生に続く第２のサイクルで、デコードがター

ゲット命令を発行し始め、これには通常通り末尾ポインタから始まってタグが割当てられる。キャンセルされたエントリにROBリタイア論理242が出会えば、これらは破棄される。ロード／ストアユニット530は、ROB240とロード／ストアセクションLSSEC530との間のLSCANCELラインを介して伝送されるLSCANCEL信号によってROBからゴーアヘッドで、待っている何らかのキャンセルを知らされる。LSCANCEL信号は、キャンセルされるべきアクセスバッファ605内の何らかの未処理のストアまたはロードミスを示す。アクセスバッファ605はFIFOとして動作して、次に古いストアはキャンセルされる命令である。ロード／ストアセクションLSSEC530およびアクセスバッファ（ストアバッファ）605として用いてもよいロード／ストアセクションおよびアクセスバッファの1つに関してのさらなる詳細は、「高性能ロード／ストア機能ユニットおよびデータキャッシュ」と題される同時係属中の米国特許出願連続番号第146，376号に記載され、その開示はここに引用によって援用される。

【0140】ある特定の命令の実行の際に例外が起これば、どのグローバルアクションも要求されない。例外状態は単に、ROB240に戻される結果状態に反映される。適切なトラップベクトル数が、一般に通常の結果オペランドの代わりに戻される（これはRF更新が禁じられないときを除き、この場合にはROBはベクトル数を発生する）。トラップベクトル数とは、様々な種類のベクトルのうちのどれが起こったか、および特定のトラップの発生の際にどこに行くべきかを示す数である。トラップの発生となる典型的な例は、0での除算、算術的オーバーフロー、およびTLBページの欠如がある。ROB240が命令を用済とする処理の際に例外状態に出会えば、これは、ROB240からのすべてのエントリをクリアし、すべての機能ユニットにEXCEPTION信号をアサートしてこれら（およびIDECODE）をクリアし、Vfビットについてトラップベクトルを発生し、フェッチャ257に処理コードをトラップするように再び指示を与えることからなるトラップ動作を始める。Vfビットは、トラップが外部フェッチとして（ベクトルテーブルからのロードとして）発生すべきか、または定数をベクトル数と連結させて内部的に発生されるべきかを示す。Vfビットは、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドのAm29000マイクロプロセッサシリーズのアーキテクチャの特徴である。

【0141】レジスタファイル235内にストアされたデータは、マイクロプロセッサの現在の実行状態を表わすことがわかる。しかしながら、ROB240にストアされたデータは、マイクロプロセッサの予測実行状態を表わす。命令が用済とされるべきとき、ROB240にストアされた対応する結果が、レジスタファイル235に送られ、それから用済とされる。

【0142】[III(E)　命令フロータイミング]命令フローのタイミングに関して、スーパースカラマイクロプロセッサ500の動作を説明するために、以下の表2が与えられる。表2は、マイクロプロセッサ500のパイプラインステージと、これらの各ステージの間に起こる重要な事象とを示す。パイプラインの段階は、表2の第1の列に挙げられる。

【0143】

【表2】

1)フェッチ	PH1	命令フェッチアドレスが形成される（フェッチPC（FPC））。
	PH2	ICACHEがアクセスされる。
2)デコード	PH1	命令ブロックがXInBでデコードするように送られる。レジスタファイルポートが割当てられ、スタックポインタの付加が行なわれる。
	PH2	命令が分類され、発行が確立される。opコード、タイプおよびオペランドタグがユニットにブロードキャストされる。レジスタファイルがアクセスされる。RA／RBフィールドがROBの内容に対してチェックされる。
3)実　行	PH1	A／Bオペランドバスが RF／ROBによって駆動されるか、またはオペランドが結果バスによって選択され得て、発行ビット（XINDISP）がアサートされる。命令が投入されるか、または待合わせステーションに留かれる。結果バスがリクエストされる。
	PH2	命令が実行される。機能ユニットがその待合わせステーションの発行のフル／空状態を信号で伝える。［分岐誤予測が決定される（PH2の遅くに）］。
4)結果転送	PH1	機能ユニットに結果バスが呼可され、結果が結果バスを介してROBに送られる（何らかのユニットへの結果バス転送のために利用可能となる）。［フェッチPC（FPC）が正しいターゲットPCで更新される］
	PH2	ROBが格納のためのエントリを調べる［分岐先に関するキャッシュアクセス］。
5)ライトバック	PH1	結果がレジスタファイルに送られライトバックされる。PC1が更新される［分岐先ブロックがデコードに送られる］。
	PH2	［分岐先ブロックはデコード中である］

【0144】表2は、機能停止のない、マイクロプロセッサ500における基本的な整数命令の流れにおいて各相（各マイクロプロセッササイクルのPH1およびPH2）で何が起こるかと分岐訂正タイミング（かっこ内）を示す。

【0145】［III（F）　メモリ管理ユニット、データキャッシュおよびバスインタフェースユニット］メモリ管理ユニット（MMU）247は、本質的には、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドによって製造されるAM29050マイクロプロセッサのものと同じである。MMU247は、命令フェッチおよびデータアクセスのために仮想アドレスを物理アドレスに変換する。AM29050とマイクロプロセッサ500との命令フェッチに関しての違いは、AM29050では、分岐先キャッシュBTCへの参照の際にMMUが調べられるが、一方、マイクロプロセッサ500は分岐先キャッシュを用いず、BTC参照のためにMMUを調べない。分岐先キャッシュは、分岐先のみのキャッシュである。分岐先キャッシュは、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドが製造するAm29050マイクロプロセッサのスカラパイプラインの一部を形成する。BTCは、1クロックサイクルにつき1度命令をフェッチする。

【0146】命令フェッチアドレス変換のためのMMU247の必要をさらになくすために、ICACHE205は、キャッシュミスの際にICACHEが参照する1エントリ高速変換バッファ（TLB）615を含む。TLBは、1エントリTLBでヒットしない変換が必要なときにリフィルされる。したがって、TLB615は、MMUからの必要に応じてリフィルされる。MMU247はICACHE205と密に結合されるわけではないので、これはリフィル時間を短縮し、MMUに対する負荷を減じる。

【0147】データキャッシュ245は、物理アドレス、2ウェイセットアソシアティブ8Kキャッシュとして構成される。この実施例では、4Kを下回るページサイズに関しては、アドレス変換がまず行なわれる。この要件は、1Kおよび2Kページサイズについて当てはまり、ヒットしているロードの待ち時間を2サイクルに増大する。しかしながら、キャッシュインデックスにおいて不確かな1ビットを有する4Kページサイズは、キャッシュを2つの4Kアレイに分割して扱わわ、これによって2つの可能なブロックへのアクセスが可能になる。4ウェイ比較が、正しいものを選択するためにMMUからの2つの物理アドレスと2つのキャッシュタグとの間で行なわれる。

【0148】データキャッシュ245は、コピーバック／ライトスルーが混合された方法をとる。より具体的には、書込ミスはライトスルーとして行なわれ、割当はなく、書込ヒットは、ロードによって前に割当てられたブロックに対してのみ起こり、キャッシュコヒーレンシーに依存してライトスルーを起こし得る。マイクロプロセ

ッサ５００は、マルチプロセッサシステムおよびＭＯＥＳＩ－モディファイド・オーンド・エクスクルーシブ・シェアード・インバリッド（フューチャーバス）プロトコルを用いるキャッシュ可能メモリの効率的なＩ／Ｏのためにデータキャッシュコヒーレンシーをサポートする。ＭＯＥＳＩプロトコルは、特定のキャッシュブロックの５つの状態のうちの１つを示す。図３ないし図５のマイクロプロセッサ５００がＭＯＥＳＩプロトコルを用いるのに対して、後述の図１０および１１に示されるマイクロプロセッサは類似したＭＥＳＩプロトコルを用いる。

【０１４９】バスインタフェースユニット（ＢＩＵ）２６０は、アドバンスト・マイクロ・ディバイシズ・インコーポレイテッドが製造するＡＭＤ２９０３０マイクロプロセッサと同じ外部インタフェースを用いる。さらに、ＢＩＵ２６０は、アドレス、命令、およびデータのために単一の内部３２ビットバス、すなわち内部アドレスデータ（ＩＡＤ）バス２５０を用いる。

【０１５０】この特定の実施例では、外部メモリとも称される主メモリ２５５は、Ｉ／Ｏとデータ／命令とのみを区別する単一の平らなスペースである。示される特定の実施例では、メモリ２５５はリードオンリーメモリ（ＲＯＭ）を含まず、命令とデータとの区別を行なわない。他のタイプの外部メモリの構成を、主メモリ２５５として用いてもよい。

【０１５１】図３ないし図５に示されるように、ＢＩＵ２６０、ＩＣＡＣＨＥ２０５、ＤＣＡＣＨＥ２４５、ＭＭＵ２４７およびＳＲＢＳＥＣ５１２は、すべて３２ビットＩＡＤバス２５０によって結合される。ＩＡＤバス２５０は、キャッシュミスおよびコヒーレンシー動作の際の外部アクセスのために、主にＢＩＵ２６０とキャッシュ（ＩＣＡＣＨＥ２０５、ＤＣＡＣＨＥ２４５）との間の通信のために用いられる。ＩＡＤバス２５０は、アドレスとデータの両方を扱う。これはスタティックバスであり、ＰＨ１の間はＢＩＵ２６０が駆動し、ＰＨ２の間は他のすべてのユニットが駆動する。ＩＡＤバス２５０に対するいかなるリクエストも、図７に示されるバス調停ブロックによって与えられるバス調停および許可を通らなくてはならない。スペースを節約するために、バス調停ブロック７００は、図３ないし図５のマイクロプロセッサ５００のブロック図には図示しない。

【０１５２】ＩＡＤバスの調停は、調停動作の中で第１の優先順位を得るバス観察（キャッシュコヒーレンシーに関して）を含む。ＩＡＤバスに対するリクエストは、ＰＨ１の早くに行なわれ、ＰＨ１の非常に遅くに応答される。機能ユニットがＰＨ１でＩＡＤバスを許可されると、後続のＰＨ２の間にアドレスをＩＡＤバスに送り、ＢＩＵによるある動作（たとえば命令フェッチ、ロード）をリクエストし得る。

【０１５３】ＩＡＤバス２５０は、外部バスおよびマイクロプロセッサ５００内のすべての主要なアレイを互いに連結する、比較的低周波数のアドレス、データおよび制御バスである。ＩＡＤバス２５０は、マッピングアレイへの特殊レジスタ更新、ＭＭＵ変換、キャッシュリフィル、バス観察等の比較的低周波数の動作の転送を与える。本発明の一実施例では、ＩＡＤバス２５０は、それにアドレスおよびデータがマルチプレクスされる３２ビットを含む。ＩＡＤバス２５０はまた、１２の制御ライン、すなわちＩＣＡＣＨＥ、ＤＣＡＣＨＥ、ＴＬＢ、ＳＲＢＳＥＣ、ＬＳＳＥＣおよびＢＩＵの各ブロックについての、それに結合される読出制御ラインおよび書込制御ラインを含む。

【０１５４】図７に示されるＩＡＤ調停ブロック７００は、どの構成要素（ＩＣＡＣＨＥ２０５、ＢＩＵ２６０、ＢＲＮＳＥＣ５２０、ＤＣＡＣＨＥ２４５、ＳＲＢＳＥＣ５１２またはＭＭＵ２４７）がある特定の時間にＩＡＤバス２５０に対してアクセスを許可されるかを決定するために、リクエスト／許可プロトコルを用いる。ＢＩＵ２６０を介して外部メモリ２５５が、バス観察の目的のために最高の優先順位を許可される。バス観察は、マイクロプロセッサ５００のためのデータ一致プロトコルの一部である。マイクロプロセッサ５００は、データキャッシュ内に局所的に保持される変更されたデータを含み得るので、このようなデータは、メモリへの書込が起こるときに更新される。マイクロプロセッサ５００はまた、データキャッシュ内に局所的に保持される変更されたブロックへの読出が起こると、変更されたデータを与える。バス観察を備えたコピーバック機構が、マイクロプロセッサ５００のキャッシュ動作において用いられる。

【０１５５】図７に示されるように、ＩＡＤ調停ブロック７００とＩＡＣＨＥ２０５、ＢＩＵ２６０、ＢＲＮＳＥＣ５２０、ＤＣＡＣＨＥ２４５、ＳＲＢＳＥＣ５１２またはＭＭＵ２４７の各々との間に、それぞれのリクエストラインが結合される。これらのリクエストラインの各々は制御論理７０５に結合され、その出力はドライバ７１０に結合される。ＩＡＤ調停ブロック７００は、ＩＣＡＣＨＥ２０５、ＢＩＵ２６０、ＢＲＮＳＥＣ５２０、ＤＣＡＣＨＥ２４５、ＳＲＢＳＥＣ５１２またはＭＭＵ２４７のためのそれぞれの許可ラインを含む。特定の構成要素がＩＡＤバス２５０へのアクセスを求めると、その構成要素はＩＡＤ調停ブロック７００と制御７０５とにリクエスト信号を送る。たとえば、ＢＩＵがメモリアクセスを行なうためにＩＡＤバス２５０へのアクセスを得たいと仮定する。この場合、ＢＩＵ２６０は、ＩＡＤ調停ブロック７００および制御７０５にＩＡＤバスアクセスリクエストを送る。ＩＡＤ調停ブロック７００は、ＩＡＤバス２５０に対するアクセスのリクエストが同時に複数存在するとき、リクエストの優先順位を決定する。調停ブロック７００は、優先順位の方式に従っ

てそれがＩＡＤバスへのアクセスを許可されるべきだと決定した特定の装置の許可ラインに許可を投入する。この例では、許可信号はＢＩＵ許可ラインに投入され、ＢＩＵ２６０はＩＡＤバス２５０へのアクセスを進める。

【０１５６】制御回路７０５の出力はＩＡＤバス２５０に結合される。以下の構成要素ＩＣＡＣＨＥ２０５、ＢＩＵ２６０、ＢＲＮＳＥＣ５２０、ＳＲＢＳＥＣ５１２、ＤＣＡＣＨＥ２４５およびＭＭＵ２４７の各々には、このような構成要素がＩＡＤバス２５０を駆動するのを可能にするドライバ回路７１０が備えられる。これらの構成要素の各々にはさらに、これらの構成要素がＩＡＤバス２５０からの値をラッチするのを可能にするラッチ７１５が備えられる。制御回路７０５は、ＩＡＤバスのためのリクエスト許可プロトコルを与える。機能ユニットは局所的に、ＩＡＤバスへのアクセスが求められていることを認め、調停ブロック７００にリクエストを送る。調停ブロック７００は最も優先順位の高いリクエストを受取り、それにしたがってアクセスを許可する。ラッチ７１５は、そのブロックに転送が起こっていれば、リクエストされたデータの読出を示す。ドライバ７１０は、局所的に利用可能な値の駆動を示し、別のブロックがそれを読出す他の何らかの位置を駆動する。ＩＡＤバス２５０へのアクセスを得るためにこのバス調停を通るとある特り時間が加わるが、それでも許容可能な性能を与えることが見いだされた。マイクロプロセッサ５００にＩＡＤバス２５０を設けることは、ＩＡＤバスに接続される上述のセクションすべての間に専用の経路を設けることよりもコスト効率がはるかに良い。

【０１５７】図８は、マイクロプロセッサ５００のパイプラインの複数の段階を通してのその選択された信号の状態を示すタイミング図である。図８は、逐次的処理のためのこのようなパイプラインを示す。対照的に、図９のタイミング図は、マイクロプロセッサ５００の同様のタイミング図ではあるが、図９のタイミング図は分岐誤予測および回復が起こる場合のものである。

【０１５８】より具体的には、図８および図９は、フェッチ、デコード、実行、結果／ＲＯＢ（結果転送－結果がＲＯＢに転送される）、用尽／レジスタファイル（ライトバック－オペランドがＲＯＢからレジスタファイルに格納される）の５つの実効パイプライン段階を通してのマイクロプロセッサ５００の動作を示す。マイクロプロセッサパイプラインの５段階は、これらのタイミング図の上部に横方向に挙げられる。これらのタイミング図を構成する信号は、図の左に縦方向に挙げられ、以下のとおりである。Ｐｈ１信号は、マイクロプロセッサ５００のクロック信号である。ＦＰＣ（３１：０）はフェッチＰＣバス（ＦＰＣ）である。ＩＲ０－３（３１：０）は命令バスを表わす。タイミング図はまた、ＲＯＢ内の特定のデコード命令が必要とする特定のオペランドを示すソースＡ／Ｂポインタを示す。タイミング図はまた、レジスタファイル／ＲＯＢアクセスを示すＲＥＧＦ／ＲＯＢアクセスを含む。Issue instr/dest tags 信号は、命令／行先タグの投入を示す。A/B read operand buses 信号は、ＡおよびＢオペランドバスを介してのＡおよびＢオペランドの転送を示す。Funct unit exec.信号は、機能ユニットでの投入された命令の実行を示す。Result bus arb信号は、結果バスに対する調停を示す。Result bus forward信号は、機能ユニットによって結果が発生された後の果バスを介しての結果の転送を示す。ROB write result信号は、結果がＲＯＢに書込まれることを示す。ROB tag forward 信号は、ＲＯＢから機能ユニットへのオペランドタグの転送を示す。REGF write/retire 信号は、ＲＯＢからレジスタファイルへの結果の格納を示す。ＰＣ（３１：０）信号は、命令がもう推論的なものではないとして用済とされると必ず更新されるプログラムカウンタ（ＰＣ）を示す。

【０１５９】図８のタイミング図では、パイプラインは逐次的な命令ストリームの実行に関して示される。この例では、予測実行経路が実際にとられ、キャッシュから直接利用可能である。簡単に言えば、フェッチパイプライン段階において、命令はマイクロプロセッサによる処理のためにキャッシュからフェッチされる。命令はデコードパイプライン段階でデコードされて、実行パイプライン段階で実行される。ソースオペランドバスおよび結果バスは、整数のサイズに対応する３２ビットの幅であることがわかる。命令バスオペランドバスが倍精度浮動小数点演算のために６４ビット値を駆動するには２サイクルが必要である。

【０１６０】結果パイプライン段階では、オペランド値が、結果を発生した機能ユニットから実行のために他の機能ユニットに直接転送される。結果段階のクロック相ＰＨ１において、推論命令の位置に、何らかの状態ともに行先結果が書込まれる。言い換えれば、機能ユニットによって発生された結果はリオーダバッファ内のエントリに置かれ、このエントリは、割当てられたタグが、タグがそのソースオペランドの１つであることを必要とする後続の命令によって検出される。これは図８のタイミング図において、図８の矢印に示されるようにソースＡ／Ｂオペランドバスへのタグ転送を介した結果「ｃ」の直接転送で示される。図８において、「ａ」および「ｂ」は結果「ｃ」をもたらすオペランドであり、「ｃ」および「ｄ」は結果「ｅ」をもたらすオペランドであることがわかる。

【０１６１】パイプラインの最後の段階である用尽パイプライン段階では、リアルプログラムカウンタ（ＰＣ）

またはリタイアPCが保持される。用尽パイプライン段階のPH1クロック相において、動作の結果はリオーダバッファからレジスタファイルに書込まれ、リタイアPCはこのライトバックを反映するように更新される。言い換えれば、リタイアPCは、もう推論的なものではないとしてレジスタファイルに格納されたばかりの命令を含むように更新される。この命令のためのエントリまたはリオーダバッファ内の結果は割当から外される。エントリが割当から外されるので、レジスタ「c」の後続の参照は、リオーダバッファからの推論的読出ではなく、レジスタファイルからの読出となる。

【0162】図9は、図8のタイミング図と同じ5パイプライン段階を示すが、図9のタイミング図は、分岐誤予測が起こるときのマイクロプロセッサ500の動作を示す。XFPCは、FPCパス信号の反転を示す。

【0163】IV．スーパースカラマイクロプロセッサの代替実施例

上述のスーパースカラマイクロプロセッサの実施例は、命令opコードがすべて同じサイズであるRISCプログラムを処理するのに最も有利に用いられるが、マイクロプロセッサ800としてこれから説明するマイクロプロセッサの実施例は、opコードのサイズが可変である命令の処理が可能である。たとえば、マイクロプロセッサ800は、可変長opコードを用いるよく知られたインテル（Intel）（登録商標）命令セットによって用いられる、いわゆるX86命令を処理することができる。マイクロプロセッサ800は、上述のマイクロプロセッサ500のRISCコアに類似したRISCコアを用いる。「RISCコア」という用語は、マイクロプロセッサ500の機能ユニット、リオーダバッファ、レジスタファイルおよび命令デコーダを含む、本質的にRISC（縮小命令セットコンピュータ）のアーキテクチャであるマイクロプロセッサ500の中核を指す。

【0164】マイクロプロセッサ800のアーキテクチャは、インテルX86命令セットに見られるようないわゆるCISC（完全命令セットコンピュータ）命令を取込み、これらの命令をRISC類似命令（ROP）に変換することができ、これらがRISCコアによって処理される。この変換プロセスは、図10および11に示されるマイクロプロセッサ800のデコーダ805で起こる。デコーダ805はCISC命令をデコードし、CISC命令をROPに変換し、ROPを実行のために機能ユニットに発行する。デコーダ805の動作および構造についてのさらなる詳細は、本譲受人に譲受された「スーパースカラ命令デコーダ」（"Superscalar Instruction Decoder"）と題される同時係属中の米国特許出願連続番号第146，383号から見いだされ、その開示はここに引用によって援用される。

【0165】マイクロプロセッサがそのRISCコアに1サイクルにつき多数の命令を供給する能力は、このスーパースカラマイクロプロセッサによって提供される著しい性能の向上の理由の1つである。命令キャッシュ（ICACHE）810は、バイトのキューまたはバイトキュー（バイトQ）815としてこの命令供給を行なう、マイクロプロセッサ800の構成要素である。本発明のこの特定の実施例では、命令キャッシュ810は16Kバイト実効4ウェイセットアソシアティブ線形アドレス命令キャッシュである。

【0166】図10および11に示されるように、命令キャッシュ810のバイトQ815は、命令デコーダ805に供給される。命令デコーダ805は、それに与えられる各命令を1つ以上のROPにマッピングする。デコーダ805のROP発行ウィンドウ820は、ICACHE810からの命令がそれにマッピングされ得る4つの発行位置を含む。4つの発行位置は、D0、D1、D2、およびD3として示される。第1の例では、デコーダ805にバイトQ815によって与えられる命令は、2つのROP発行位置にマッピングされ得る命令であると仮定する。この場合、この第1の命令がデコーダ805に与えられると、デコーダ805は命令を発行位置D0に与えられる第1のROPと、発行位置D1に与えられる第2のROPとにマッピングする。後続の第2の命令が3つのROP位置にマッピング可能であると仮定する。この第2の命令がデコーダ805にバイトQ815によって与えられると、命令は発行位置D2に与えられる第3のROPと、発行位置D3に与えられる第4のROPとにマッピングされる。発行位置D0ないしD3にあるROPは機能ユニットに発行される。第2の命令がマッピングされる、残っている第3のROPは、このようなROPが発行され得る前に次の発行ウィンドウが処理されるのを待たなくてはならないことがわかる。

【0167】命令キャッシュ810がどの特定のバイトをバイトQ815に送るかに関する情報は、命令キャッシュ810の入力である分岐予測ブロック825に含まれる。分岐予測ブロック825は、ブロック単位で次に予測された分岐位置を示す次ブロックアレイである。分岐予測機能ユニット835は、図3ないし図5に示されるマイクロプロセッサ500のBRNSEC520と類似した態様で、分岐を実行する。命令キャッシュ810にはまた、外部メモリからリクエストされた命令キャッシュミスをフェッチするプリフェッチャブロック830が備えられる。

【0168】マイクロプロセッサ800は、デコーダ805の4つのROP位置がそれに投入され得る4つの整数機能ユニット、すなわち分岐機能ユニット835、ALU0／シフタ機能ユニット840、ALU1機能ユニット845、および特殊レジスタ機能ユニット850を含む。分岐機能ユニット835は、1クロックサイクルにつき1つの新しいROPが分岐機能ユニット835によって受入れられるように、1サイクルの待ち時間を有

する。分岐ユニット８３５は２エントリ待合わせステーション８３５Ｒを含む。本明細書の目的のため、２エントリを含む待合わせステーションは、２つの待合わせステーションと同じであると考えられる。分岐機能ユニット８３５は、すべてのＸ８６分岐、コールおよびリターン命令を扱う。これはまた条件付分岐ルーチンを扱う。

【０１６９】ＡＬＵＯ／シフタ機能ユニット８４０は、１サイクルの待ち時間を示す。１クロックサイクルにつき１つの新しいＲＯＰがユニット８４０に受入れられる。ＡＬＵＯ／シフタ機能ユニット８４０は、２つまでの推論ＲＯＰを保持する２エントリ待合わせステーション８４０Ｒを含む。すべてのＸ８６算術および論理計算は、この機能ユニットまたはその代わりに他方の算術論理装置ＡＬＵ１　８４５に渡る。さらに、シフトローテートまたはファインドファーストワンのような命令は、ＡＬＵＯ／シフタ機能ユニット８４０に与えられる。

【０１７０】ＡＬＵ１機能ユニット８４５もまた、１サイクルの待ち時間を示す。１クロックサイクルにつき１の新しいＲＯＰがＡＬＵ１機能ユニット８４５によって受入れられることがわかる。ＡＬＵ１機能ユニットは、２つまでの推論ＲＯＰを保持する２エントリ待合わせステーション８４５Ｒを含む。すべてのＸ８６算術および論理計算は、この機能ユニットかまたは他方の算術論理装置ＡＬＵＯに渡る。ＡＬＵＯおよびＡＬＵ１は、１サイクルにつき２つまでの整数結果演算を計算することを可能にする。

【０１７１】特殊レジスタ機能ユニット８５０は、Ｘ８６レジスタファイル８５５の外にある内部制御、ステータスおよびマッピング状態を扱うための特殊ブロックである。本発明の一実施例では、特殊レジスタ機能ユニット８５０は、ＲＯＰが特殊レジスタ機能ユニット８５０に投入されるときに未処理である推論状態がないので、待合わせステーションを持たない。特殊レジスタブロック８５０は、その構造および機能の点で、上述の特殊レジスタブロック５１２と類似している。

【０１７２】ロード／ストア機能ユニット８６０および浮動小数点機能ユニット８６５は、デコーダ８０５のＲＯＰ発行ウィンドウ８２０に結合される。ロード／ストア機能ユニット８６０は、複数エントリ待合わせステーション８６０Ｒを含む。浮動小数点機能ユニット８６５は２つの待合わせステーション８６５Ｒを含む。データキャッシュ８７０が、データのストアおよびそのための検索を与えるために、ロード／ストア機能ユニット８６０に結合される。浮動小数点機能ユニット８６５は、４１ビット整数／浮動小数点演算混在バス８７５および結果バス８８０に連結される。より詳細には、オペランドバス８７５は、４１ビット幅を示す８つの読出オペランドバスを含む。結果バス８８０は、４１ビット幅を示す５つの結果バスを含む。浮動小数点ユニットの整数／浮動小数点混在オペランドおよび結果バスへの連結によって、推論整数および浮動小数点ＲＯＰの両方のために、１つのレジスタファイル８５５および１つのリオーダバッファ８８５を用いることが可能になる。２つのＲＯＰは８０ビット拡張精度演算を形成し、これは浮動小数点待合わせステーション８６５Ｒから浮動小数点機能８６５内の８０ビット浮動小数点コアに入力される。

【０１７３】浮動小数点機能ユニット８６５の８０ビット浮動小数点コアは、浮動小数点加算器、浮動小数点乗算器、および浮動小数点除算／平方根機能ユニットを含む。浮動小数点ユニット８６５内の浮動小数点加算器機能ユニットは、２サイクルの待ち時間を示す。浮動小数点加算器は、８０ビットの拡張結果を計算し、これが転送される。浮動小数点乗算器は、拡張精度乗算演算のために６サイクルの待ち時間を示す。３２Ｘ３２乗算器が、単精度乗算演算のために用いられる。浮動小数点機能ユニット８６５内の３２Ｘ３２乗算器は、拡張精度を必要とする６４ビット仮数演算のためにマルチサイクル化される。浮動小数点除算／平方根機能ユニットは、６４ビット仮数を２ビット／クロックで計算するために基数－４対話型除算を用いる。

【０１７４】Ａ／Ｂオペランドバスのバス幅が４１ビットであるこの実施例では、整数ユニットに延びるＡ／Ｂオペランドバスに関して、３２ビットがオペランド専用であり、残りの９ビットが制御情報専用であることが認められる。Ａ／Ｂオペランドバスのバス幅が４１ビットではなく、３２ビットまたは他のサイズである、本発明の他の実施例も企図されることに注目されたい。このような３２ビットオペランドバス幅の構成では、オペランドバスから分離される制御ラインが、制御情報の伝送のために用いられる。

【０１７５】ロードストア機能ユニット８６０は、４エントリ待合わせステーション８６０Ｒを含む。ロードストア機能ユニット８６０は、２つのロードまたはストア動作が１サイクルについて投入されることを可能にする。ロードストアセクションはまた、線形アドレスを計算し、メモリのリクエストされたセグメントへのアクセス権をチェックする。データキャッシュ８７０内のヒット／ミスのチェックに関してのロードまたはストア動作の待ち時間は１サイクルである。２つまでのロード動作が、同時にデータキャッシュ８７０にアクセスし、その動作を結果バス８８０に送ることができる。ロードストアセクション８６０は、整数および浮動小数点ロードおよびストア動作の両方を扱う。

【０１７６】図１０および１１に示されるように、マイクロプロセッサ８００は、リオーダバッファ８８５に結合されるレジスタファイル８５５を含む。レジスタファイル８５５およびリオーダバッファ８８５の両方が、オペランド振分回路８９０を介してオペランドバス８７５に結合される。レジスタファイル８５５、リオーダバッファ８８５およびオペランド振分回路８９０は協働し

て、オペランドを機能ユニットに与える。結果が機能ユニットから得られると、これらの結果はリオーダバッファ８８５に送られ、その中のエントリとしてストアされる。

【０１７７】より詳細には、レジスタファイル８５５およびリオーダバッファ８８５は、プログラム実行の間のオペランドのためのストアを与える。レジスタファイル８５５は、整数および浮動小数点命令の両方のためのマッピングされたＸ８６レジスタを含む。レジスタファイルは、中間計算を保持するための、ならびに整数および浮動小数点の一時レジスタを含む。本発明のこの特定の実施例では、レジスタファイル８５５内のすべてのレジスタは、８つの読出および４つの書込ラッチとして実現される。このように設けられた４つの書込ポートによって、１クロックについて２つまでのレジスタファイル行先が書込まれることを可能にする。これは、１ポートについて１つの整数値であるか、またはレジスタファイルに浮動小数点結果が書込まれている場合には、１ポートにつき浮動小数点値の半分であってもよい。８つの読出ポートによって、２つのソース読出動作を伴う４つのＲＯＰの各々が、１クロックサイクルについて扱入されることが可能になる。

【０１７８】リオーダバッファ８８５は、１６までの推論ＲＯＰのキューを保持する、１６エントリ環状ＦＩＦＯとして構成される。リオーダバッファ８８５はしたがって、１６のエントリを割当てることができ、その各々が整数結果または浮動小数点結果の半分を含むことができる。リオーダバッファ８８５は、１クロックサイクルにつき４つのＲＯＰを割当てることができ、１クロックサイクルにつき５までのＲＯＰを確立し、１クロックサイクルにつき４つまでのＲＯＰをレジスタファイル８５５に格納することができる。マイクロプロセッサ８００の現在の推論状態は、必要に応じて後続の転送のためにリオーダバッファ８８５内に保持される。リオーダバッファ８８５はまた、各エントリについて各ＲＯＰの相対順序を示す状態を維持する。リオーダバッファ８８５はまた、割込またはトラップルーチンによる処理のためにミスしている分岐および例外をマークする。

【０１７９】リオーダバッファ８８５は、８つのオペランドでそれぞれ８つのオペランドバス８７５を駆動できる。リオーダバッファ８８５は、５つの結果バス８８０を介して１サイクルにつき５つまでの結果を受取ることができる。オペランドバスは８つの４１ビット整数／浮動小数点共通バスであることが認められる。８つのオペランドバスは、デコーダ８０５のＲＯＰ発行ウィンドウ８２０内の４つのＲＯＰ発行位置に対応する。４つのＲＯＰ発行位置の各々は、ソースＡオペランドおよびソースＢオペランドを有することができる。このように形成される４つのＡおよびＢ読出オペランド対の各々は、ＲＯＰ発行ウィンドウ８２０内の固定ＲＯＰおよびソース読出位置専用である。

【０１８０】レジスタファイル８５５およびリオーダバッファ８８５は、読出オペランドバス８７５を駆動するマイクロプロセッサ８００内の装置である。デコードされたＲＯＰに関して推論の行先がなければ、すなわちＲＯＰによってリクエストされたオペランドがリオーダバッファになければ、レジスタファイルがそのオペランドを供給する。しかしながら、推論の行先が存在すれば、すなわちデコードされたＲＯＰによってリクエストされたオペランドがリオーダバッファ内にあれば、そのオペランドのためのリオーダバッファ内の最も新しいエントリが、対応するレジスタの代わりに機能ユニットに送られる。このリオーダバッファ結果値は、これがもしリオーダバッファ内に存在すれば推論結果であるか、または機能ユニット内でまだ完了されていない推論の行先に関するリオーダバッファタグでもあり得る。

【０１８１】５つの結果バス８８０は４１ビットバスである。読出オペランドおよび結果バスは、すべての整数機能ユニットの入力および出力であることがわかる。これらの同じ読出オペランドおよび結果バスはまた、浮動小数点機能ユニット８６５の浮動小数点待合わせステーション８６５Ｒの入力および出力である。浮動小数点待合わせステーション８６５Ｒは、４１ビットオペランドおよび結果バスを、必要であればその構成する専用機能ユニットに送る８０ビット拡張精度バスに変換する。

【０１８２】マイクロプロセッサ８００の整数および浮動小数点機能ユニットには、これらのユニットの待合わせステーションを介してＲＯＰの局所バッファ処理が与えられる。これらの機能ユニットのほとんどで、局所バッファ処理は、ＦＩＦＯとして構成される２エントリ待合わせステーションの形をとる。このような待合わせステーションの目的は、デコーダ８０５の発行論理が、機能ユニットに推論ＲＯＰを、このような推論ＲＯＰのソースオペランドが現在利用可能であるかどうかに関わらず、送ることを可能にすることである。本発明のこの実施例では、したがって、長い計算またはロードが完了するのを待つことなく、多数の推論ＲＯＰ（１６まで）が投入され得る。この態様で、はるかに高い命令レベルの並列性が与えられ、マイクロプロセッサ８００は、そのピーク性能に近く動作することが可能になる。

【０１８３】待合わせステーションの各エントリは、２つのソースオペランドまたはタグと、各エントリに関連するｏｐコードおよび行先に関しての情報を保持することができる。待合わせステーションはまた、リオーダバッファが未処理であるとマークしたソースオペランド結果（リオーダバッファがオペランド自体ではなくオペランドタグを与えることによってそれについてマークしたオペランド）を、このような結果を待っている他の機能ユニットに直接送ることができる。本発明のこの特定の実施例では、機能ユニットの待合わせステーションは、

典型的には1クロックサイクルにつき新しいエントリを1つ受入れ、1サイクルにつき1つの新しいエントリを機能ユニットに送ることができる。

【0184】これに対する例外は、その待合わせステーションから1クロックサイクルにつき2つのエントリを受入れ、かつ用済とすることができるロード／ストアセクション860である。ロード／ストアセクション860はまた、4つのエントリのより深い待合わせステーションFIFOを有する。

【0185】すべての待合わせステーションのエントリは、例外が起こるようなことがあれば、1クロックサイクル以内に割当から外されることができる。分岐誤予測が起こると、中間結果が機能ユニットから流し出され、リオーダバッファからの割当から外される。

【0186】マイクロプロセッサ800は、プリフェッチユニット830を介して命令キャッシュ810に、およびバスインタフェースユニット900に結合される内部アドレスデータバス895を含む。バスインタフェースユニット900は、主メモリまたは外部メモリ（図示せず）に結合され、そのためマイクロプロセッサ800には外部メモリアクセスが与えられる。IADバス895はまた、図10および11に示されるように、ロード／ストア機能ユニット860に結合される。

【0187】データキャッシュ870は、ロード／ストアユニット860に結合される。本発明のある特定的な実施例では、データキャッシュ870は、8Kバイト、線形アドレス、2ウェイセットアソシアティブ、デュアルアクセスキャッシュである。アドレスおよびデータラインは、図示されるようにデータキャッシュ870をロード／ストア機能ユニット860に結合する。より具体的には、データキャッシュ870は、キャッシュ870とロード／ストアユニット860との間の2つの組のアドレスおよびデータ経路を含み、ロード／ストア機能ユニット860からの2つの同時アクセスを可能にする。これらの2つのアクセスは、16バイトデータキャッシュラインサイズに整列される、8ないし32ビットロードまたはストアアクセスであってもよい。データキャッシュ870は、16バイトラインまたはブロックに構成される。この特定的な実施例では、データキャッシュ870は線形にアドレスされるか、またはセグメントベースのアドレスからアクセスされ、ページテーブルベースの物理アドレスではない。データキャッシュ870は4つのバンクを含み、これらは、データキャッシュ内の1つのラインが4つのバンクの各々における4つのバイトを有するように構成される。したがって、2つのアクセスのビット［3：2］の線形アドレスが同じでないかぎり、2つのアクセスは同時にキャッシュ870内のデータアレイにアクセスすることができる。

【0188】データキャッシュ870は、2ウェイアソシアティブである。これは、クロックの相PH1において2つの線形アドレスをとり、その4つのバンクにアクセスする。その結果としてのロード動作は、後続のクロック相PH2で完了し、結果バスのうちの1つを駆動することができる。機能ユニットによる結果バスのリクエストは、結果をライトバックしようとする他の機能ユニットからのリクエストと調停される。

【0189】命令キャッシュ810およびデータキャッシュ870は、それぞれの命令キャッシュ線形タグアレイおよびデータキャッシュ線形タグアレイを含み、これらのキャッシュにストアされたデータエントリおよび命令のアドレスに対応する。図10および11に示されるように、マイクロプロセッサ800はまた、命令キャッシュ810およびデータキャッシュ870内のそれぞれ命令およびデータの物理アドレスを追跡するためにIADバス895に結合される物理タグI／Dブロック910を含む。より具体的には、物理タグI／Dブロック910は、これらのキャッシュの物理アドレスを維持する物理命令／データタグアレイを含む。ブロック910の物理命令タグアレイは、命令キャッシュ810の対応する線形命令タグアレイに関する構成を反映する。同様に、ブロック910内の物理データタグアレイの構成は、命令キャッシュ810内の対応する線形データタグアレイの構成を反映する。

【0190】物理I／Dタグは、命令キャッシュタグであるかデータキャッシュタグであるかに依存して、有効、共有、および変更ビットを有する。データキャッシュ物理タグがセットされた変更ビットを有する場合には、これはリクエストされたデータエレメントが、線形データキャッシュ内の等価な位置にあることを示す。マイクロプロセッサ800は外部メモリへのバックオフサイクルを開始し、リクエストされた変更ブロックを、リクエストしている装置がそれを後で見ることができるメモリに書込む。

【0191】高速変換バッファ（TLB915）が、図示のようにIADバス895と物理タグI／Dブロック910との間に結合される。TLB915は、128の線形－物理ページ変換アドレスおよび128までの4Kバイトページのためのページ複写をストアする。この高速変換バッファアレイは、ランダムな置換えを有する4ウェイセットアソシアティブ構造として構成される。TLB915は、X86アーキテクチャのために規定される線形－物理アドレス変換機構を扱う。この機構は、最も最近の線形－物理アドレス変換のキャッシュを用いて、有効な変換のために外部ページテーブルを探すのを防ぐ。

【0192】バスインタフェースユニット900は、IADバス895をメモリ等の外部装置にインタフェースさせる。IADバス895は、マイクロプロセッサ800の様々な構成要素を接続するのに用いられるグローバル64ビット共有アドレス／データ／制御バスである。

ＩＡＤバス８９５は、キャッシュブロックリフィル、ライトアウト変更ブロックのため、ならびに特殊レジスタユニット８５０、ロード／ストア機能ユニット８６０、データキャッシュ８７０、命令キャッシュ８１０、物理Ｉ／Ｄタグブロック９１０、高速変換バッファ９１５、およびバスインタフェースユニット９００等の機能ブロックにデータおよび制御情報を渡すために用いられる。

【０１９３】Ｖ．代替実施例の動作概説

ＣＩＳＣプログラムが実行されるとき、ＣＩＳＣプログラムの命令およびデータが、これらの命令およびデータをストアするのに用いられた何らかの記憶媒体から主メモリにロードされる。一旦、バスインタフェースユニット９００に結合される主メモリにプログラムがロードされると、命令はプログラム順にデコーダ８０５に、機能ユニットによる発行および処理のためにフェッチされる。より具体的には、デコーダ８０５によって１度に４つの命令がデコードされる。命令は、主メモリからバスインタフェースユニット９００に、ＩＡＤバス８９５を介して、プリフェッチユニット８３０を通り、命令キャッシュ８１０に、そしてデコーダ８０５に流れる。命令キャッシュ８１０は、デコーダ８０５によってデコードされて発行されるべき命令の保管場所として機能する。命令キャッシュ８１０は、分岐予測ユニット８３５と関連して動作し、デコーダ８０５に、推論的に実行されるべき次の予測された命令ブロックである、４命令幅の命令ブロックを与える。

【０１９４】より具体的には、命令キャッシュ８１０は、主メモリからバスインタフェースユニット９００を介してフェッチされた命令ブロックを含む、ＩＣＳＴＯＲＥと示されるストアアレイを含む。ＩＣＡＣＨＥ８１０は、１６バイトラインまたはブロックに構成される、１６Ｋバイト実効線形アドレス命令キャッシュである。各キャッシュラインまたはブロックは、１６のＸ８６バイトを含む。各ラインまたはブロックはまた、各バイトについて５ビットプリデコード状態を含む。ＩＣＡＣＨＥ８１０は、命令デコーダ８０５に次に予測されたＸ８６命令バイトをフェッチする役目を果たす。

【０１９５】ＩＣＡＣＨＥ８１０は、ＦＥＴＣＨＰＣ（ＦＰＣ）と示される推論プログラムカウンタを維持する。この推論プログラムカウンタＦＥＴＣＨＰＣは、キャッシュ情報を維持する以下の３つの別個のランダムアクセスメモリ（ＲＡＭ）アレイにアクセスするために用いられる。より詳細には、キャッシュ情報を含む３つの上述のＲＡＭアレイは、１）ストアアレイＩＣＳＴＯＲＥ内の対応するブロックに関するバイト有効ビットおよび線形タグを維持するアレイであるＩＣＴＡＧＶを含む。キャッシュ内の各エントリは、１６バイト有効ビットおよび２０ビット線形タグを含む。この特定の実施例では、２５６のタグが用いられる。２）アレイＩＣＮＸＴＢＬＫは、ストアアレイＩＣＳＴＯＲＥ内の対応するブロックに関する分岐予測情報を維持する。ＩＣＮＸＴＢＬＫアレイは、各々が１６Ｋバイト実効Ｘ８６命令に対応する、２５６エントリの４つの組に構成される。この次ブロックアレイ内の各エントリは、シーケンシャルビット、最後に予測されたバイトおよびサクセッサインデックスから構成される。３）ＩＣＳＴＯＲＥアレイは、Ｘ８６命令バイトと５ビットのプリデコード状態とを含む。プリデコード状態は、各バイトと関連し、特定のバイトがマッピングされるＲＯＰの数を示す。このプリデコード情報は、命令のデコードを、これらがデコーダ８０５に与えられると速める。バイトキューまたはＩＣＢＹＴＥＱ８１５は、プリフェッチユニット８３０によってＩＣＡＣＨＥ８１０に与えられる命令プリフェッチストリームの現在の推論状態を与える。ＩＣＡＣＨＥ８１０として用いることができる命令キャッシュに関するより多くの情報は、同時係属中で本譲受人に譲受された、「可変バイト長命令に特に適した推論命令キューおよびそのための方法」と題する米国特許連続出願番号第１４５，９０２号に記載され、その開示がここに引用によって援用される。

【０１９６】デコーダ８０５（ＩＤＥＣＯＤＥ）は、マイクロプロセッサ８００内の命令デコードおよび発行動作を実行する。より具体的には、デコーダ８０５は、デコード１およびデコード２と称するマイクロプロセッサパイプラインの２つの段階を実行する。デコード１の初めの間、プリフェッチされ、予測実行されたバイトはバイトキューの指定された充満位置に送られる。これらのバイトは次に、バイトキュー８１５内の独立バイトと併合される。デコード２パイプラインステージにおいて、リオーダバッファのエントリが、次のクロック相で投入され得る対応するＲＯＰに割当てられる。

【０１９７】デコーダ８０５は、バイトキュー８１５から未処理のＸ８６命令バイトおよびプリデコード情報を取入れ、これらをＲＯＰ発行ユニット８２０内の４つのＲＯＰ位置に割当てる。デコーダ８０５は、どの特定の機能ユニットに各ＲＯＰが伝送されるべきかを決定する。デコーダ８０５として用いることができるデコードの１つのより詳細な説明は、ディビッド・ビィ・ウィットおよびマイケル・ディ・ゴダード（Dabid B. Witt and Michael D. Goddard）による「スーパースカラ命令デコーダ」と題される米国特許出願連続番号第１４６，３８３号に記載され、その開示をここに引用によって援用する。ＩＣＡＣＨＥおよびデコーダ回路によって、マイクロプロセッサ８００は、１クロックサイクルにつき４つのＲＯＰをデコードし、ＲＩＳＣ類似データ経路に送ることができる。４つのＲＯＰは、機能ユニットに発行され、これが結果をリオーダバッファ８８５と、これらの結果を必要とする他の機能ユニットとに送る。

【０１９８】レジスタファイル８５５およびリオーダバッファ８８５は、プログラムの流れにおける命令に推論

実行を与えるようにともに動作する。マイクロプロセッサ８００の整数コア、レジスタファイル８５５、リオーダバッファ８８５のより詳細な説明を、図１２を参照して行なう。マイクロプロセッサ８００の整数コアは、整数コア９２０として示され、分岐予測ユニット８３５、ＡＬＵ０、ＡＬＵ１、および特殊レジスタ８６０を含む。

【０１９９】この特定の実施例において、レジスタファイル８５５は、１２の３２ビットレジスタ（整数レジスタ）と２４の４１ビットレジスタ（浮動小数点レジスタ）として構成される。これらのレジスタは、デコーダ８０５から並列して４つまでのＲＯＰに関してアクセスされる。デコーダ８０５によって与えられるレジスタファイルポインタは、どの特定のレジスタが特定のＲＯＰにおけるオペランド値としてリクエストされるか、およびアクセスのサイズを決定する。

【０２００】レジスタファイル８５５はマイクロプロセッサ８００のアーキテクチャ状態を含む一方で、リオーダバッファ８８５はマイクロプロセッサ８００の推論状態を含むことが記められる。レジスタファイル８５５のタイミングは、８つまでの並列読出ポインタで、デコーダ２パイプラインステージの相ＰＨ２でアクセスされるようにされる。これらの８つまでの読出ポインタの受取に応答して、レジスタファイル８５５は、このように選択されたオペランド値を、後続のクロックＰＨ１相で対応するオペランドバスに送る。

【０２０１】リオーダバッファ８８５をレジスタファイル８５５に結合する不能化バスが図１２に示される。不能化バスは８ライン幅であり、リクエストされた読出値がリオーダバッファ８８５内の推論エントリとして見いだされたことを示す８つの無効信号を含む。この例では、レジスタファイル８５５は無効にされ、リクエストされた読出オペランド値をオペランドバスに置くことを許されない。その代わりに、推論エントリがリオーダバッファ８８５内に存在するので、リオーダバッファ８８５は、リクエストされた実際のオペランド値か、またはその値に関するオペランドタグを与える。

【０２０２】リオーダバッファ８８５は、この特定の実施例では１６のエントリを含み、推論ＲＯＰ結果値のキューとして動作する。図１３により詳細に示されるように、リオーダバッファ８８５は、キューの先頭および末尾に対応する２つのポインタ、すなわち先頭ポインタおよび末尾ポインタを含む。キューの割当の発行されるＲＯＰへのシフトは、これらのポインタを増分または減分することによって起こる。

【０２０３】リオーダバッファ８８５に与えられる入力は、デコーダ８０５がそこで割当てようとするＲＯＰの数（１ブロックにつき４つまでのＲＯＰ）、これらの４つのＲＯＰのためのソースオペランドポインタ値、およびそれぞれの行先ポインタ値を含む。リオーダバッファ８８５は次に、その現在の推論キューからこれらのエントリを割当てようとする。エントリスペースが発行されるＲＯＰのために利用可能であれば、エントリは末尾ポインタの後に割当てられる。

【０２０４】より具体的には、エントリがデコーダ８０５からリクエストされると、キューの先頭から次のエントリが割当てられる。特定のエントリの数は、デコーダ８０５からのその特定のＲＯＰに関する行先タグとなる。行先タグは、実行されるべき特定の命令とともに、対応するＲＯＰ位置で機能ユニットに送られる。「４ＲＯＰ行先タグ」と示される専用行先タグバスは、図１２において、リオーダバッファ８８５から整数コア９２０の機能ユニットへ、およびマイクロプロセッサ８００の残りの機能ユニットへの出力として示される。機能ユニットはこのように、実行されるべき各ＲＯＰに関する行先情報を与えられ、そのため機能ユニットは効果的に結果バスを介してＲＯＰの結果がどこに送られるはずであるかを知る。

【０２０５】上述のことより、推論実行された結果値またはオペランドは、このような結果オペランドがもはや推論ではなくなるまで、リオーダバッファ８８５内に一時的にストアされることが記められる。可能性のあるオペランド値のプールは、したがってリオーダバッファによって与えられ、デコーダ８０５によって与えられてデコードされる後続のＲＯＰによって用いられる。

【０２０６】リオーダバッファ８８５内にエントリが存在するときには、元のレジスタ番号（すなわちＥＡＸ）が、特定のＲＯＰ結果に関して割当てられたリオーダバッファエントリ内に保持される。図１３は、先頭および末尾ポインタの間の推論状態にあるエントリを、これらのエントリ内の縦の破線で示す。各リオーダバッファエントリは、その元の行先レジスタ番号に参照し戻される。ＲＯＰ発行ユニット８２０の４つのＲＯＰ位置からの８つの読出ポインタ値のうちの何らかのものがエントリに関連する元のレジスタ番号に一致すると、そのエントリの結果データが、有効であれば転送され、またはそのエントリに関連する動作がまだ機能ユニットで未処理であればタグが転送される。

【０２０７】リオーダバッファ８８５は、デコード８０５によって発行された新しいＲＯＰの正しい推論状態を、これらのＲＯＰをプログラム順に割当てることで維持する。４つのＲＯＰはその現在の位置からリオーダバッファキューの末尾位置まで、それらの読出オペランドのいずれかにおける一致を探しながらスキャンする。特定のリオーダバッファエントリにおいて一致が起これば、レジスタファイル８５５内の対応する読出ポートが不能化され、実際の結果オペランドまたはオペランドタグが、適切な機能ユニットによって受取られるようにオペランドバスに与えられる。この構成によって、動作に影響を与えることなく、リオーダバッファに存在する同

じレジスタの複数の更新を可能にする。結果転送がこのように達成される。

【０２０８】図１３に示されるように、リオーダバッファ８８５は、リオーダバッファキューまたはアレイ９３０にストアされた結果オペランドの用尽を制御するリタイア論理９２５を含む。キュー９３０に格納された結果オペランドがもはや推論でなければ、このような結果オペランドはリタイア論理制御のもとでレジスタファイル８５５に転送される。これを起こすためには、ＲＯＰの格納をインタフェースするリタイア論理、レジスタファイルへのライトバック、最後の４つのＲＯＰエントリの状態がスキャンされる。リタイア論理９２５は、割当てられたＲＯＰエントリのうちのいくつが有効な結果を現在有しているかを決定する。リタイア論理はまた、これらのＲＯＰエントリのうちのいくつが、ライトバックのないＲＯＰに対して、レジスタファイルへのライトバック結果を有するかをチェックする。さらに、リタイア論理は、発生される分岐、ストアおよびロードミスについてスキャンする。完全な命令が最後の４つのＲＯＰ内に存在すれば、このようなＲＯＰはレジスタファイルに格納される。しかしながら、ＲＯＰエントリをスキャンする間に、特定のＲＯＰにおいて例外が起こったことを示す状態が見いだされれば、その後のすべてのＲＯＰが無効にされ、トラップベクトルフェッチリクエストが、ＲＯＰエントリに格納された例外状態情報により形成される。

【０２０９】さらに、リオーダバッファ内のＲＯＰをスキャンしている際に分岐誤予測状態に出会えば、誤予測された経路にあるとしてマークされなかった最初のＲＯＰに出会うまで、ＥＩＰレジスタの更新またはライトバックなく、リタイア論理はこれらのＲＯＰエントリを無効にする。リタイア論理９２５（図１３参照）内に含まれるＥＩＰレジスタ（図示せず）は、推論的ではない実行された命令を推論で実行された命令から分ける、実行下のプログラムにおけるロールする分解点を表わすリタイアＰＣまたはプログラムカウンタを保持する。ＥＩＰまたはリタイアＰＣは、リオーダバッファ８８５からレジスタファイル８５５への結果オペランドの格納の際に、このように格納された命令がもはや推論的ではないことを反映するように、継続的に更新される。リオーダバッファ８８５は推論状態を素早く追跡し、１クロックサイクルにつき複数のＸ８６命令またはＲＯＰを用済とすることができることが認められる。マイクロプロセッサ８００は、例外条件または分岐誤予測に出会えば、迅速に無効とし、正しい命令ストリームをフェッチし始めることができる。

【０２１０】マイクロプロセッサ８００の機能ユニットの一般的な構成を、ここで図１４に例示的な目的のために示される一般化された機能ユニットブロック図を参照して説明する。ｏｐコード、Ａオペランド、Ｂオペランド、および行先タグを含むＲＯＰは、図９の一般化された機能ユニットに発行されていることを思い起こされたい。図１４の最も左の部分には、それに発行される命令から特定のＡオペランドを選択する（１：４）Ａオペランドマルチプレクサ９３２に４つのＡオペランドバスが与えられることが認められる。同様の態様で、４つのＢオペランドバスが、図１４の機能ユニットが実行すべき対象の命令のための特定のＢオペランドを選択する（１：４）Ｂオペランドマルチプレクサ９３５に結合される。４つの行先／ｏｐコードバスが、この機能ユニットによって実行されている特定の命令のためのｏｐコードおよび行先タグを選択するマルチプレクサ９４０に結合される。

【０２１１】この機能ユニットは、マルチプレクサ９４０への「ファインドファーストＦＵＮＣタイプ」入力でタイプバスをモニタする。より特定的には、機能ユニットは、その機能ユニットのタイプに一致する第１のＲＯＰを探し、１：４マルチプレクサ９３２、９３５、および９４０を可能化して、対応するオペランドおよびタグ情報を図１４の機能ユニットの待合わせステーション１に送る。たとえば、実行ユニット９４５が算術論理装置１（ＡＬＵ）であり、かつマルチプレクサ９４０のＴＹＰＥ入力で機能ユニットに与えられる命令タイプがＡＤＤ命令であると仮定すると、発行された命令の行先タグ、ｏｐコード、Ａオペランド、およびＢオペランドが、選択マルチプレクサ９３２、９３５および９４０を介して待合わせステーション１に送られる。

【０２１２】第２の待合わせステーション、すなわち待合わせステーション０が、待合わせステーション１と実行ユニット９４５との間に認められる。図１４の機能ユニットは、このように２つの待合わせステーションを含むと言われ、または待合わせステーションは２つのエントリを保持することができると言う。この２エントリ待合わせステーションは、最も古いエントリが待合わせ０として示されるＦＩＦＯとして実現される。待合わせステーション０および１は、レジスタファイル８５５またはリオーダバッファ８８５のいずれかからオペランドバスを介して機能ユニットに何が送られたかに依存して、オペランドまたはオペランドタグのいずれかを保持することができる。

【０２１３】その結果を５つの結果バスに与える他の機能ユニットからの結果の転送を達成するために、機能ユニットは、Ａ転送論理９５０およびＢ転送論理９５５を有する。転送論理９５０は、ソースＡオペランドに一致するタグを求めて５つの結果バスをスキャンし、一致が起これば、Ａ転送論理９５０は、対応する結果バスを待合わせステーション１のＡデータ部分９６０に送る。実際のＡオペランドではなくＡオペランドタグがマルチプレクサ９３２を介して送られると、Ａオペランドタグは、Ａタグ９６５と示される位置にストアされることに

注目されたい。一致を求めて5つの結果バスにおいてスキャンされる結果タグと比較されるのは、Aタグ位置965にストアされたAオペランドタグである。同様の態様で、B転送論理955は、Bオペランドタグ位置970にストアされたBオペランドタグに一致する何らかの結果タグに関して5つの結果バスをスキャンする。一致が見いだされれば、対応する結果オペランドが結果バスから検索され、Bデータ位置975にストアされる。機能ユニットによって実行されているROPのopコードおよび行先タグは、タグおよびopコード位置980にストアされる。

【0214】ROP命令を実行するのに必要なすべての情報が機能ユニット内で集められれば、ROP命令は実行のために実行ユニット945に投入される。より具体的には、Aオペランドおよびオペランドが、待合わせステーションによって実行ユニット945に送られる。その命令のためのopコードおよび行先タグが、タグおよびopコード位置980によって実行ユニット945に送られる。実行ユニットは命令を実行し、結果を発生する。実行ユニットは次に、アービトレータ（図示せず）に結果リクエスト信号を送ることで結果バスへのアクセスに対して調停する。実行ユニット945が結果バスへのアクセスを許可されると、結果許可信号がアービトレータから実行ユニット945によって受取られる。実行ユニット945はその結果を指定された結果バスに置く。

【0215】この結果と同じタグを有する未処理のオペランドを有する他の機能ユニットに結果が転送される。結果はまた、実行されたROPの行先タグと関連するエントリでそこにストアするためにリオーダバッファ885にも与えられる。

【0216】実用において、機能ユニットは、命令が実行している間結果バスに対して調停する。より具体的には、機能ユニットに有効エントリが存在するとき、すなわち実行のために必要なすべてのオペランド、opコード、および行先タグ情報が集められたとき、命令は実行ユニット945に投入され、実行ユニット945が実際にその命令を実行している間、機能ユニットは結果バスに対して調停する。各待合わせステーションが行先タグとともに局所opコードのための記憶機構を含むことが認められる。このタグは、結果パイプラインステージの間にROPが最終的にライトバックする位置を示す。この行先タグはまた、待合わせステーション内の各エントリと保持され、そのFIFOを介して押される。

【0217】一般化された機能ユニットブロック図を図14に関して説明したが、実行ユニット945は、分岐予測ユニット835、ALU0／シフタ840、ALU1845、ロード／ストア860、浮動小数点ユニット865および特殊レジスタ850のいずれであってもよく、これらの特定の機能に関する適切な変更を加えてもよい。

【0218】特定の機能ユニットへの結果バスの許可が行なわれると、結果値が結果バスに送られ、待合わせステーション内の対応するエントリがクリアされる。結果バスは、41ビットの結果と、行先タグと、通常、有効および例外等の状態指示情報とを含む。マイクロプロセッサ800のパイプライン化された動作において、上述の機能ユニットの動作のタイミングは、実行段階の間に起こる。クロック相PH1の間、オペランド、行先タグおよびopコードは、ROPが発行され、待合わせステーションに置かれる際に送られる。PH2クロック相の間、opコードによって説明される動作は、すべてのオペランドの準備ができていれば実行され、実行の間、機能ユニットは値をリオーダバッファに送返すために結果バスに対して調停する。

【0219】図15は、分岐機能ユニット835のより詳細な図である。分岐機能ユニット835は、ジャンプ命令ならびにより複雑なコールおよびリターンマイクロルーチンを含む非逐次的フェッチをすべて扱う。分岐ユニット835は、待合わせステーション835Rと、予測発生分岐を追跡するための分岐FIFO980を含む。分岐機能ユニット835はまた、加算器985と、インクリメンタ990と、分岐予測コンパレータ995とを含み、これらはすべてPC相対分岐を扱うためのものである。

【0220】分岐機能ユニット835は、図15に示される分岐予測発生FIFO980を用いて推論分岐を制御する。より具体的には、命令キャッシュ810によって予測されたすべての非順次的フェッチは、分岐予測FIFO980に送られ、その分岐のPC（プログラムカウンタ）とともにそこでラッチされる。この情報は、ターゲットバス（XTARGET）およびデコードPCバスに送られて、分岐機能ユニットに渡る。対応する分岐が後にデコードされ、投入されると、予測情報、オフセット、および分岐のPCが、分岐機能ユニット835によって局所的に計算される。一致が起これば、この結果はターゲットPCと一致を示す状態とともに、リオーダバッファ885に正しく送り返される。分岐誤予測が起これば、正しいターゲットが、フェッチを始めるために命令キャッシュ810へ送られ、またミスしている予測された分岐に含まれる後続のROPをキャンセルためにリオーダバッファ885へ送られる。この態様で、実行は正しいターゲットPCで再び始めることができ、このようにして実行プロセスの失敗を防ぐ。誤予測が起こると必ず、分岐機能ユニット835は、新しいターゲットアドレスとインデックスとの両方を、予測情報があったブロックに送り、このアレイを更新する。このことは、マイクロプロセッサが、予測アレイ情報を更新しながら同時に、命令の新しく正しいストリームをフェッチし始めることを意味する。マイクロプロセッサはまた、新し

いブロックで予測情報にアクセスして、どのバイトが予測実行されるかを知ることに注目されたい。ＩＣＮＸＴＢＬＫアレイは、予測情報がその第２のポートを介して更新され得るように、デュアルポートである。誤予測が起こるブロックからの予測情報は、逐次／非逐次、分岐位置、およびキャッシュアレイ内の予測実行される第１のバイトの位置等の情報である。

【０２２１】加算器９８５およびインクリメンタ９９０は、現在の分岐命令の現在のＰＣ＋オフセット、および逐次的であれば次のＰＣの命令長＋ＰＣを局所的に計算する。これらの値は、コンパレータ９９５によって、局所分岐発生キュー（ＦＩＦＯ９８０）内の予測発生分岐と比較されて、このような分岐を予測する。

【０２２２】ここで、マイクロプロセッサ８００の動作をそのパイプラインステージを通して示すタイミング図を説明する前に、マイクロプロセッサ８００の主な内部バスを概略的に説明する。バスラインの先頭のＸは、一方の相でダイナミックにチャージされ、他方の相で条件付でアサートされる偽バスを示す。マイクロプロセッサ８００の内部バスは以下のものを含む。

【０２２３】ＦＰＣ（３１：０）－Ｐｈ１、スタティック。このフェッチＰＣバスは、命令キャッシュ８１０からバイトキュー８１５への推論命令プリフェッチのために用いられる。ＦＰＣバスは、図３ないし図５のマイクロプロセッサ５００のＦＰＣブロック２０７と実質的に同じ機能を果たす、ＩＣＡＣＨＥ８１０内のＦＰＣブロック８１３に結合される。

【０２２４】ＸＴＡＲＧＥＴ（４１：０）－Ｐｈ１、ダイナミック。このバスは、誤予測分岐および例外を指示しなおすためにターゲットＰＣを命令キャッシュおよび分岐予測ユニット（８２５／８３５）に送る。

【０２２５】ＸＩＣＢＹＴＥｎＢ（１２：０）－Ｐｈ１、ダイナミック。このバスは、現在リクエストされているプリフェッチＸ８６命令および対応するプリデコード情報の命令キャッシュストアアレイＩＣＳＴＯＲＥの出力である。この特定の実施例では、サイクルにつき全部で１６のバイトが、次に予測実行されたバイトがバイトキューの第１のオープンバイト位置を充満するように整列されてアサートすることができる。

【０２２６】ＢＹＴＥＱｎ（７：０）－Ｐｈ１、スタティック。これは、命令キャッシュからフリフェッチされた予測実行Ｘ８６命令バイトのキューを示す。この特定の実施例では、全部で１６のバイトがデコーダ８０５のデコード経路に送られる。各バイトは、ｏｐコード位置、プリフィックスバイト、ならびに命令開始および終了位置に関しての命令キャッシュからのプリデコード情報を含む。各Ｘ８６命令のＲＯＰサイズもまた、プリデコード情報に含まれる。各バイトに加えられるプリデコード情報は、バイトキュー内の１バイトについて全部で６ビットのストアを表わし、すなわち１有効ビット＋５つのプリデコードビットを表わす。

【０２２７】ＩＡＤ（６３：０）－Ｐｈ１、ダイナミック。ＩＡＤバス８９５は、主なマイクロプロセッサ８００のブロックのための一般的な相互接続バスである。これは、このようなブロック間と、外部メモリへの、およびそこからのアドレス、データ、および制御転送のために用いられ、図１０および１１に示されるとおりである。

【０２２８】ＸＲＤｎＡＢ（４０：０）－Ｐｈ１、ダイナミック。この符号は、機能ユニットに与えられる各ＲＯＰのためのソースオペランドＡバスを表わし、オペランドバス８７５内に含まれる。より具体的には、これはＲＯＰ０ないしＲＯＰ３のための全部で４つの４１ビットバスを含む。オペランドバスには含まれる対応するタグバスは、リオーダバッファ８８５からの実際のオペランドデータの代わりに、リオーダバッファ８８５からの転送されたタグが存在することを示す。

【０２２９】ＸＲＤｎＢＢ（４０：０）－Ｐｈ１、ダイナミック。この符号は、機能ユニットに送られる各ＲＯＰのためのソースオペランドＢバスを示す。このバス構造は、ＲＯＰ０ないしＲＯＰ３のための４つの４１ビットバスを含み、８つの読出オペランドバス８７５内に含まれる。対応するタグバスは、リオーダバッファ８８５からの実際のオペランドデータの代わりに、転送されたオペランドタグがこのバスに存在することを示すことがやはり認められる。

【０２３０】ＸＲＥＳｎＢ（４０：０）－Ｐｈ１、ダイナミック。この符号は、８、１６、３２ビット整数、または８０ビット拡張結果の１／２のための結果バス８８０を示す。対応するタグおよび状態バス８８２は、この結果バスでエントリを確立することがわかる。

【０２３１】マイクロプロセッサ８００は、フェッチ、デコード１、デコード２、実行、結果／ＲＯＢおよび用足／レジスタファイルの段階を含む６段階パイプラインを含む。明瞭にするために、デコードステージは図１６においてデコード１およびデコード２に分割されている。図１６は、逐次的な実行が行なわれているときのマイクロプロセッサパイプラインを示す。連続するパイプライン段階は、図１６の縦方向の列で表わされる。マイクロプロセッサ８００において選択された信号は、パイプラインの種々の段階で現われることを横方向の列で表わす。

【０２３２】図１６の逐次実行パイプライン図は、以下の選択された信号を表わす。「Ｐｈ１」は、システムクロック信号の前縁を表わす。システムクロック信号は、Ｐｈ１およびＰｈ２成分の両方を含む。

【０２３３】「ＦＰＣ（３１：０）」は、バイトキュー８１５からのフェッチＰＣバスを表わす。

【０２３４】「ＩＣＢＹＴＥｎＢ（１２：０）」は、バイトキュー８１５に結合される命令キャッシュ８１０の

ＩＣＳＴＯＲＥアレイからのＩＣＢＹＴＥバスである。【０２３５】「ＢＹＴＥＱｎ（７：０）」は、バイトキューバスである。「ＲＯＰｍｕｘ（３：０）」は、命令ブロックおよびプリデコード情報がデコーダに与えられていることを示すデコーダ信号である。【０２３６】「Source A/B pointers 」は、デコーダ８０５によってリオーダバッファ８１５に与えられるＡおよびＢオペランドのための読出／書込ポインタである。図１０および１１には明確に図示されないが、ソースポインタは、デコードブロックからレジスタファイルおよびリオーダバッファの両方への入力であるレジスタファイル値である。【０２３７】「REGF/ROB access 」は、機能ユニットへの伝送のためにオペランド値を得るためのレジスタファイルおよびリオーダバッファへのアクセスを示す。【０２３８】「Issue ROPs/dest tags」は、デコーダ８０５による機能ユニットへのＲＯＰおよび行先タグの投入を示す。【０２３９】「A/B read oper buses 」は、機能ユニットによる、そのためのＡおよびＢオペランドまたはタグを得るためのＡおよびＢオペランドバスの読出を示す。【０２４０】「Funct unit exec 」は、機能ユニットによる実行を示す。図１６および図１７において、符号ａ＆ｂ→ｃおよびｃ＆ｄ→ｅおよびｃ＆ｇ→は、任意の演算を表わし、「ソース１オペランド、ソース２オペランド→行先」の形である。より具体的には、示されるソースレジスタは、レジスタ、すなわち一時またはマッピングＸ８６レジスタである。ａ＆ｂ→ｃの例では、「ｃ」の値は行先を表わし、結果バスおよびリオーグバッファから、予測実行ストリームの次の参照への局所的な転送を示す。【０２４１】「Result Bus arb」は、結果をリオーダバッファ、およびこの結果に対応するオペランドタグを保持しているためにその結果を必要とするかもしれない他の何らかの機能ユニットに伝送するために、結果バス８８０へのアクセスを調停している時間を示す。【０２４２】「Result Bus forward」は、結果がある機能ユニットからこの結果を未処理のオペランドとして必要としている他の機能ユニットに転送している時間を示す。【０２４３】「ROB write result」は、機能ユニットからの結果がリオーダバッファに書込まれている時間を示す。【０２４４】「ROB tag forward 」は、リオーダバッファが機能ユニットに、現在まだ結果が出ていないオペランドの代わりにオペランドタグを転送している時間を示す。【０２４５】「REGF write/retire 」は、結果がリオーダバッファのＦＩＦＯキューからレジスタファイルに格納されている時間を示す。【０２４６】「ＥＩＰ（３１：０）」はリタイアＰＣ値を示す。割込リターンは遅延分岐を持たないので、マイクロプロセッサは、おずか１つのＰＣで割込リターンの際に再始動できる。リタイアＰＣ値またはＥＩＰは、リオーグバッファ８８５のリタイア論理９２５内に含まれる。ＥＩＰは、マイクロプロセッサ５００に関して既に説明したリタイアＰＣと類似している。リタイア論理９２５は、マイクロプロセッサ５００のリタイア論理２４２に類似した機能を果たす。【０２４７】図１６のタイミング図は、Ｘ８６バイトの逐次的ストリームを実行しているマイクロプロセッサ８００を示す。この例では、予測実行経路が実際に行なわれ、また命令キャッシュから直接利用可能である。【０２４８】命令処理の第１の段階は、命令フェッチである。図示のとおり、このクロックサイクルは命令キャッシュの動作を行なうのに費やされる。命令キャッシュ８１０は、クロックサイクルのＰｈ１の間に新しいフェッチＰＣ（ＦＰＣ）を形成し、第２のクロックサイクルにおいて命令キャッシュのキャッシュアレイにアクセスする。フェッチＰＣプログラムカウンタ（タイミング図ではＦＰＣ（３１：０）として示される）は、ストアアレイと並列して線形命令キャッシュのタグアレイにアクセスする。フェッチのクロック相Ｐｈ２の遅い時点で、線形タグがフェッチＰＣ線形アドレスに一致するかどうかの決定がなされる。一致が起これば、予測実行されるバイトはバイトキュー８１５に転送される。【０２４９】命令キャッシュ内のタグおよびストアアレイにアクセスするのに加えて、フェッチＰＣはまたブロック予測アレイＩＣＮＸＴＢＬＫにアクセスする。このブロック予測アレイは、どのＸ８６バイトが予測実行されるかを識別し、次の予測実行されるブロックが逐次的であるか非逐次的であるかを識別する。Ｐｈ２でアクセスされるこの情報は、現在フェッチされているブロックのどのバイトがバイトキュー８１５に有効バイトとして送られるかを決定する。【０２５０】バイトキュー８１５は、前にフェッチされているが機能ユニットにまだ投入されておらずそこにストアされたＸ８６バイトを現在有しているかもしれない。この場合には、バイト充満位置が命令キャッシュ８１０に示されて、第１の予測バイトをこの量だけシフトして、より古いＸ８６バイトの後を充満する。【０２５１】フェッチのクロック相Ｐｈ２で分岐予測情報が起こるので、プリフェッチユニット８３０によってプリフェッチされるべき次のブロックは逐次的であっても非逐次的であってもよい、というのはどちらの場合にも、キャッシュアレイに再びアクセスするのに１クロックサイクルあるからである。したがって、分岐予測アレイによって、ブロック外の分岐が、次の逐次的ブロックにアクセスするのと同じ相対的性能を有することができ、性能の向上を与える。

【０２５２】デコード１／デコード２パイプライン段階を次に説明する。デコード１の初めに、プリフェッチされ、予測実行されたバイトが、指定された充満位置でバイトキュー８１５に送られる。これは図１６のタイミング図にＩＣＢＹＴＥｎＢ（１２：０）として示され、デコード１のＰｈ１でアサートする。これらのバイトは、バイトキュー内の何らかの未処理のバイトと併合される。バイトキューはプリデコード状態の５つのビットと、未処理のＸ８６バイトとを含み、命令の境界がどこにあるかを示す。バイトキューの先頭は、次に予測実行されたＸ８６命令の初めにある。デコード１のクロック相Ｐｈ１の中程で、命令キャッシュからの次のバイトのストリームが、バイトキュー８１５内の既存のバイトと併合され、併合されたストリームがスキャンのためにデコーダ８０５に与えられる。デコーダ８０５は、各命令がとるＲＯＰの数、および対応するＲＯＰ投入位置Ｄ０、Ｄ１、Ｄ２、およびＤ３とｏｐコードの整列を可能にするようにｏｐコードの位置を決定し、ここでＤ０にあるＲＯＰが投入すべき次のＲＯＰである。デコーダ８０５は、バイトキュー８１５内の各Ｘ８６命令のプログラムカウンタＰＣのコピーを、命令の境界間のバイト数をカウントするか、または命令キャッシュ内の分岐を検出して、その位置からフェッチされた第１のＸ８６バイトにターゲットＰＣ値を付けることによって維持する。

【０２５３】ｏｐコードおよびＲＯＰ位置付け情報、ならびにバイトキュー８１５にストアされた即値フィールドを用いることで、デコーダ８０５はデコード１のクロック相Ｐｈ２およびデコード２のクロック相Ｐｈ１の間に以下の情報をスタティックに決定する。すなわち、１）機能ユニット行先、２）ソースＡ／Ｂおよび行先オペランドポインタ値、３）ソースおよび行先動作のサイズ、および４）もしあれば、即値アドレスおよびデータ値である。デコード２のクロック相Ｐｈ１の終わりに、すべてのレジスタ読出および書込ポインタが解決され、動作が決定される。これは図１６のタイミング図でソースＡ／Ｂポインタ値のアサートによって示される。

【０２５４】図１６のタイミング図に示されるデコード２パイプライン段階において、リオーダバッファエントリは、次のクロック相で投入され得る対応するＲＯＰに割当てられる。したがって、４つまでの付加的なＲＯＰが、デコード２のＰｈ１クロック相の間に１６エントリリオーダバッファ８８５内のエントリを割当てられる。デコード２のＰｈ２クロック相の間、割当てられたすべてのＲＯＰに関するソース読出ポインタが、リオーダバッファに含まれる推論ＲＯＰのキューにアクセスしながら、同時にレジスタファイルから読出される。レジスタファイルおよびリオーダバッファアレイの両方のこの同時アクセスによって、マイクロプロセッサ８００は、実際のレジスタファイル値を用いるか、またはリオーダバッファからオペランドもしくはオペランドタグを転送するかを後で選択することができる。Ｐｈ１においてリオーダバッファ内の４つのＲＯＰエントリをまず割当て、次にＰｈ２でリオーダバッファをスキャンすることによって、まだ推論状態にあるすべての前のＲＯＰと発行されている現在のＲＯＰについて読出の従属性をマイクロプロセッサ８００は同時に探すことができる。これは、図１６のタイミング図に、ＲＥＧＦ／ＲＯＢアクセスおよびタグのチェックによって示される。

【０２５５】実行パイプライン段階において、ＲＯＰは、専用ｏｐコードバスおよび読出オペランドバスによって機能ユニットに投入される。専用ｏｐコードバスは、ＲＯＰのｏｐコードを機能ユニットに送り、一方、読出オペランドバスはオペランドまたはオペランドタグをこのような機能ユニットに伝送する。オペランドバスがオペランドを機能ユニットに送っている間の時間は、図１６のタイミング図では符号「A/B read operand buses」によって示される。

【０２５６】実行パイプライン段階のＰｈ１クロック相の後半で、機能ユニットはこのような機能ユニットにどのＲＯＰが投入されたか、およびこのような機能ユニット内の局所待合わせステーションから何らかの未処理のＲＯＰの投入準備ができているかを判断する。待合わせステーション内に含まれる最も古い命令が最初に実行されることが確実になるように、機能ユニットの待合わせステーションでＦＩＦＯが維持されることに注目されたい。

【０２５７】命令が機能ユニット内で実行準備ができている場合には、実行パイプライン段階のＰｈ１の遅くにこのような実行を始め、この段階のＰｈ２にわたってスタティックに続く。Ｐｈ２の終わりに、機能ユニットは、１６の結果バスＲＯＢ信号によって示されるように５つの結果バスのうちの１つに対して調停する。言い換えれば、結果バス調停信号がこの時間の間にアサートされる。機能ユニットが結果バスへのアクセスを許可されると、これは後続のＰｈ１で割当てられた結果バスを駆動する。

【０２５８】図１６のタイミング図で示される結果バイパライン段階は、結果をある機能ユニットからこのような結果を必要としている別のものへと転送することを示す。結果バイプライン段階のクロック相Ｐｈ１において、推論ＲＯＰの位置は、行先結果および何らかの状態を伴ってリオーダバッファに書込まれる。リオーダバッファ内のこのエントリは、割当てられたとともに有効であるという指示を与えられる。一旦割当てられたエントリがこのように確立されると、リオーダバッファは、リクエストされた読出アクセスの受取の際に、オペランドタグではなくオペランドデータを直接転送することができる。結果バイプライン段階のクロック相Ｐｈ２において、新しく割当てられたタグが、そのソースオペランドの１つとしてこれを要求する後続のＲＯＰによって検出

され得る。これは、図16のタイミング図において、「ROB tag forward 」を介してソースA／Bオペランドバスへの結果Cの直接転送として示される。

【0259】用尽パイプライン段階は、図16のタイミング図のパイプラインの最終段階である。この段階は、ＥＩＰレジスタの形での真のプログラムカウンタ（リタイアＰＣ）が維持され、バス指示ＥＩＰ（31：0）によって示されるように更新される段階である。図16に示されるように、ＥＩＰ（31：0）のタイミング図は、リオーダバッファからレジスタファイルへの命令の格納の際に、新しいＰＣ（またはリタイアＰＣ）が発生されるところを示す。リオーダバッファからレジスタファイルへの結果の格納の実際の動作は、図16の「REGF write/retier 」と符号を付される信号によって示される。図16において、用尽パイプライン段階のクロック相Ｐｈ1において、動作の結果はレジスタファイルに書込まれ、ＥＩＰレジスタはこの命令がもう実行されたことを反映するように更新される。リオーダバッファ内の対応するエントリは、値がリオーダバッファからレジスタファイルへと書込まれるのと同じクロック相Ｐｈ1において割当から外される。リオーダバッファ内のこのエントリが割当から外されたので、レジスタCへの後続の参照は、リオーダバッファからの推論読出ではなく、レジスタファイルからの読出となる。この態様で、マイクロプロセッサのアーキテクチャ状態が真に反映される。

【0260】図17は、分岐誤予測の際のプロセッサ800のタイミング図である。図17のタイミング図は、以下を除いては図16のタイミング図と同じ信号タイプを示す。

【0261】ＢＲＮ_ＭＩＳＰ信号は、分岐誤予測が起こったときを示す。ＸＴＡＲＧＥＴ（31：0）信号は、予測されたターゲット分岐命令が分岐ユニット835に送られるときを示す。

【0262】図17のタイミング図は、分岐誤予測および回復の間のマイクロプロセッサ800のパイプラインの段階を示す。このタイミング図は、第1のサイクルが分岐の実行サイクルであり、かつ後続のサイクルが予測の訂正および新しい命令ストリームのフェッチに関わると仮定する。この特定の実施例において、誤予測された分岐命令の実行の完了から正しい経路の実行の開始まで3サイクルの遅延が存在することが認められる。

【0263】図17に示されるパイプラインのフェッチ段階は、ＸＴＡＲＧＥＴ（31：0）バスが、命令キャッシュ810に予測されたターゲットに関しての情報を与えるために、分岐機能ユニット835から命令キャッシュ810に駆動されることを除いては、図16の通常のフェッチ段階に類似している。分岐機能ユニットは、分岐誤予測が実際に起こったことを判断する、マイクロプロセッサ800のブロックであることが認められる。分岐機能ユニットはまた、正しいターゲットを計算する。このターゲットは、結果バス880を介して誤予測状態指示とともに結果がリオーダバッファに戻されるのと同じときに送られる。結果バスはまた、真の分岐が起こった場合に分岐命令を用涼とする際にＥＩＰレジスタを更新するための正しいＰＣ値を含む。ＸＴＡＲＧＥＴバスは、フェッチされたＰＣバスに駆動され、命令キャッシュアレイがアクセスされる。ヒットが起これば、バイトは前と同様にバイトキューに送られる。

【0264】誤予測が起これば、バイトキュー815内のすべてのバイトは、信号ＢＲＮ_ＭＩＳＰのアサートで、フェッチの第1の相において自動的にクリアされる。訂正された経路がフェッチされ、デコードされるまでは、さらなるＲＯＰはデコーダ805から発行されない。

【0265】誤予測の結果状態がリオーダバッファにフェッチパイプライン段階のクロック相Ｐｈ1において戻されるとき、誤予測状態指示が誤予測の後のすべての推論ＲＯＰに送られ、そのためこれらはレジスタファイルまたはメモリに書込を許されない。これらの命令が次に用涼とされるべきとき、リオーダバッファ内のこれらのエントリは割当から外されて、さらなるＲＯＰが投入されることを可能にする。

【0266】分岐誤予測の間のデコード1パイプライン段階に関して、訂正された経路をデコードするための経路の残りは、命令キャッシュ810のＩＣＮＸＴＢＬＫアレイにおける予測情報の更新を除いて、逐次的なフェッチの場合と同じである。分岐の正しい方向が、予測アレイＩＣＮＸＴＢＬＫの分岐が誤予測されたその中のキャッシュブロックに書込まれる。

【0267】誤予測の間のパイプライン段階デコード2、実行、結果、用涼は、図16で戴論したものと実質的に同じである。

【0268】ＶＩ．結論－スーパースカラ高性能特徴

マイクロプロセッサによって実行されるコードから実質的な並列性を引出すことで、本発明のマイクロプロセッサにおいて高性能が達成される。命令タグ付与、待合わせステーション、転送を伴う結果バスによって、オペランドハザードが無関係の命令の実行を妨げることを防ぐ。マイクロプロセッサのリオーダバッファ（ＲＯＢ）は多数の利点を達成する。ＲＯＢは一種のレジスタ再指定を用いて、行先としての同じレジスタの異なる使用を区別し、そうでなければこれは並列性を損なってしまう恐れがある。リオーダバッファにストアされたデータはマイクロプロセッサの予測実行状態を表わし、一方レジスタファイルにストアされたデータはマイクロプロセッサの現在の実行状態を表わす。さらに、リオーダバッファは割込の際のプログラムの逐次的状態を守る。さらに、リオーダバッファは、未解決の条件付分岐を越える実行を許可することによりさらなる並列性を可能にする。並列性はさらに、高いバンド幅の命令フェッチを与

えるオンボードの命令キャッシュ（ＩＣＡＣＨＥ）によって、分岐の影響を最小にする分岐予測によって、そしてロードおよびストア動作に関する待ち時間を最小にするオンボードのデータキャッシュ（ＤＣＡＣＨＥ）によってさらに促進される。

【０２６９】本発明のスーパースカラプロセッサは、いくつかの構成要素を共有することによってダイの空間を効率的に利用して、性能を向上する。より具体的には、マイクロプロセッサの整数ユニットおよび浮動小数点ユニットは、共通の、共有データ処理バス上にある。これらの機能ユニットは、同じデータ処理バスにやはり結合される複数の特合わせステーションを含む。整数および浮動小数点機能ユニットは、データ処理バス上の共通の分岐ユニットを共有する。さらに、整数および浮動小数点機能ユニットは、共通デコーダおよび共通ロード／ストアユニット５３０を共有する。内部アドレスデータ（ＩＡＤ）バスは、本発明のマイクロプロセッサのいくつかの構成要素間での局所的通信を与える。

【０２７０】本発明のある好ましい特徴のみを、例示するために示したが、多くの変更および変形が起こるであろう。したがって、前掲の特許請求の範囲は本発明の真の精神に包含されるすべての変更および変形を含むと意図されることを理解されたい。

【図面の簡単な説明】

【図１】従来のスーパースカラマイクロプロセッサを示すブロック図である。

【図２】本発明の高性能スーパースカラマイクロプロセッサの一実施例の簡略化されたブロック図である。

【図３】本発明の高性能スーパースカラマイクロプロセッサの別の実施例の一部のより詳細なブロック図である。

【図４】本発明の高性能スーパースカラマイクロプロセッサの別の実施例の一部のより詳細なブロック図である。

【図５】本発明の高性能スーパースカラマイクロプロセッサの別の実施例の一部のより詳細なブロック図である。

【図６】結果バスに対して調停している際に機能ユニットが受ける優先順位を表わす図である。

【図７】本発明のマイクロプロセッサにおける内部アドレスデータバス調停構成のブロック図である。

【図８】図３ないし図５のマイクロプロセッサの、逐次処理の間のそのパイプラインの複数の段階を通してのタイミング図である。

【図９】図８のタイミング図と類似しているが、分岐誤予測および回復が起こる際のタイミング図である。

【図１０】本発明のスーパースカラマイクロプロセッサの別の実施例のブロック図の一部である。

【図１１】本発明のスーパースカラマイクロプロセッサの別の実施例のブロック図の一部である。

【図１２】図１０および図１１のマイクロプロセッサのレジスタファイル、リオーダバッファおよび整数コアのブロック図である。

【図１３】図１２のリオーダバッファのより詳細なブロック図である。

【図１４】図１０および図１１のマイクロプロセッサが用いる一般化された機能ユニットのブロック図である。

【図１５】図１０および図１１のマイクロプロセッサが用いる分岐機能ユニットのブロック図である。

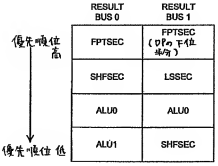
【図１６】逐次実行の間の図１０および図１１のマイクロプロセッサの動作のタイミング図である。

【図１７】分岐誤予測および回復の間の図１０および図１１のマイクロプロセッサの動作のタイミング図である。

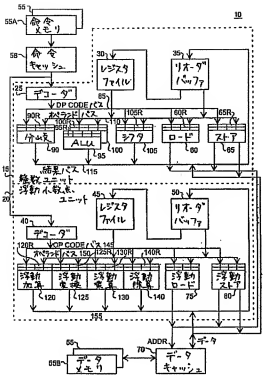
【符号の説明】

２００　マイクロプロセッサ
２０５　命令キャッシュ
２１０　命令デコーダ
２１５　整数コア
２２５　浮動小数点コア
２３５　レジスタファイル
２４０　リオーダバッファ

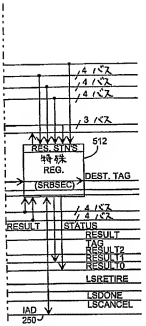
【図６】



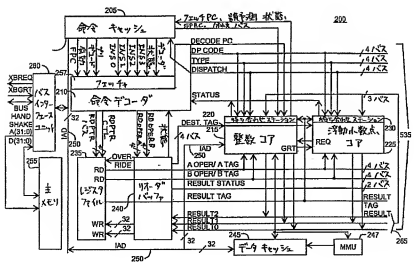
【図1】



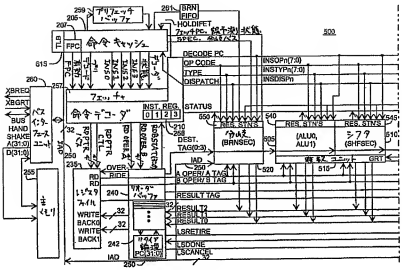
【図4】



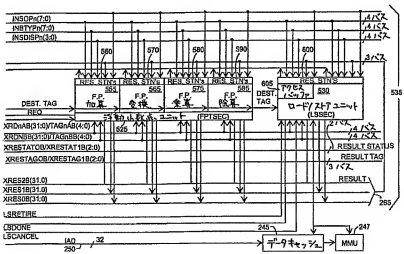
【図2】



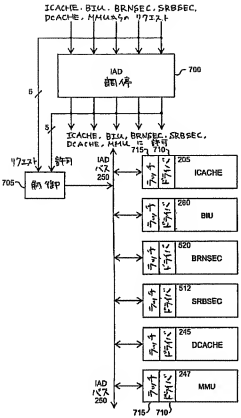
【図3】



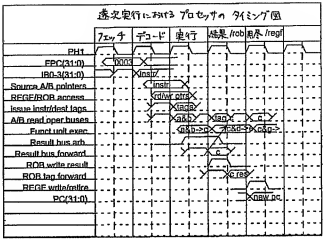
【図5】



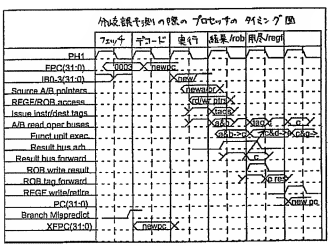
【図7】


ICACHE, BIU, BRNSEC, SRBSEC,
DCACHE, MMUからのリクエスト
IAD
調停部
700
6
5
ICACHE, BIU, BRNSEC, SRBSEC,
DCACHE, MMUへの許可
715
710
リクエスト
許可
IAD
バス
250
705
調停部
フェッチ
インタフェース
205
ICACHE
280
BIU
820
BRNSEC
812
SRBSEC
245
DCACHE
247
MMU
715
710

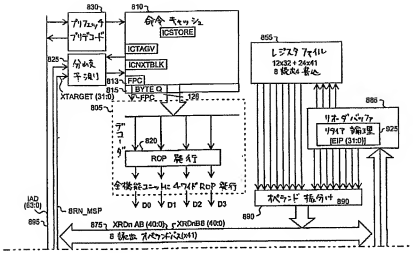
【図8】


通常実行におけるプロセッサのタイミング図
フェッチ
デコード
実行
結果/rob
用済/regf
PH1
FPC(31:0)
0005
IB0-3(31:0)
insIV
insIr
Source A/B pointers
d/wr ptrs
REGF/ROB access
d/wr data
issue instr/dest tags
a&b
a&b
c
A/B read oper buses
a&b->a
a&b->b
a->a
b->b
Funct unit exec
Result bus arb
c
Result bus forward
c
ROB write result
c res
ROB tag forward
REGF write/retire
nxw pc
PC(31:0)

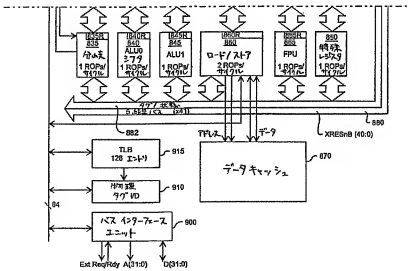
【図9】


分岐誤予測の際のプロセッサのタイミング図
フェッチ
デコード
実行
結果/rob
用済/regf
PH1
FPC(31:0)
0005
newPC
IB0-3(31:0)
newV
Source A/B pointers
d/wr ptrs
REGF/ROB access
issue instr/dest tags
A/B read oper buses
a&b
a&b
c
a&b->a
a&b->b
a->a
b->b
Funct unit exec
Result bus arb
c
Result bus forward
c
ROB write result
c res
ROB tag forward
REGF write/retire
nxw pc
PC(31:0)
Branch Mispredict
newPC
XFPC(31:0)

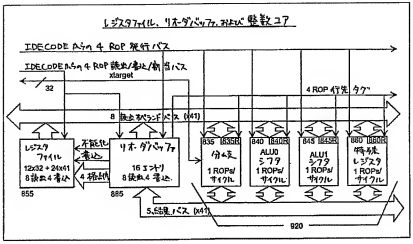
【図１０】


830
プリフェッチ
プリデコード
810
命令 キャッシュ
ICSTORE
ICTAGV
ICNXTBLK
825
分岐 予測
815
FPC
BYTE Q
XTARGET (31:0)
835
デコード
820
ROP 発行
全機能ユニットに4つのROP発行
D0
D1
D2
D3
IAD (63:0)
BRN_MSP
895
875
XRDn AB (40:0), XRDnBB (40:0)
B 読出し オペランド (63:0)
850
レジスタ ファイル
12x32 + 24x41
8 読出し 4 書込
885
リオーダ バッファ
リタイア 予測値
EIP (31:0)
925
オペランド 転送ユニット
890

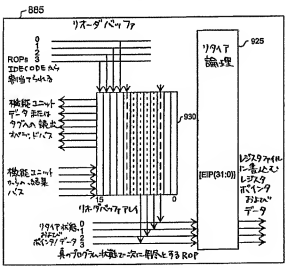
【図１１】


IEXEC
835
分岐 実
1 ROPs/サイクル
IDIDX
840
ALU0
ユニット
1 ROPs/サイクル
IEXEC
845
ALU1
1 ROPs/サイクル
IMOBX
802
ロード/ストア
2 ROPs/サイクル
FEXEC
855
FPU
1 ROPs/サイクル
850
特殊
レジスタ
1 ROPs/サイクル
リタイア 読出し
5 読出し ポート
882
XRESnB (40:0)
TLB
128 エントリ
915
データ キャッシュ
870
タグ 更新
タグ I/O
910
84
バス インターフェース
ユニット
900
Ext Req/Rdy
A(31:0)
D(31:0)

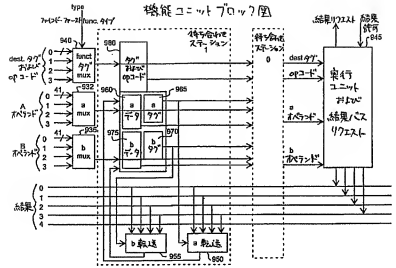
【図12】


レジスタファイル、リオーダバッファ、および機能ユニット
IDECODE Asm 4 ROP 発行バス
IDECODE Asm 4 ROP 読出/書込/衝当バス
xtarget
32
4 ROP 打消 タグ
8 読出ワンドバス (x41)
レジスタファイル
12x32 + 24x01
8読出4書込
不能化
書込
4 掃討
リオ・ダバッファ
16 エントリ
8読出4書込
855
885
835 B35R
分岐
1 ROPs/サイクル
840 B40R
ALU0
シフタ
1 ROPs/サイクル
845 B45R
ALU1
シフタ
1 ROPs/サイクル
880 B80R
特殊
レジスタ
1 ROPs/サイクル
S.結果バス (x41)
920

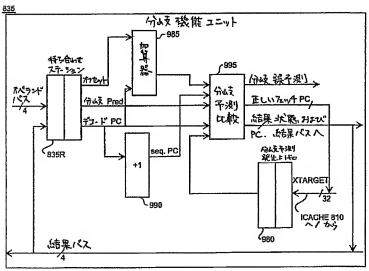
【図13】


885
リオーダバッファ
0
1
2
3
ROPs
IDECODE Asm
割当てられた
機能ユニット
データ読出
ラブへの読出
オペランドバス
機能ユニット
からの結果
バス
15
0
リオーダバッファアレイ
リタイア
論理
925
930
[EIP(31:0)]
レジスタファイル
に書込むべき
レジスタ
ポインタ
アドレス
データ
リタイア対象の
次の
ポインタ/データ
最もプログラム順序で次に実行する ROP

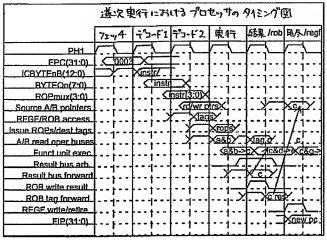
【図14】


機能ユニットブロック図
type
フィールド・フォーマット/func. タイプ
940
dest. タグ/オペコード
A オペランド
B オペランド
結果
0
1
2
3
4
func/タグ/オペコード mux
a mux
b mux
932
935
960
965
970
975
a データ
a タグ
b データ
b タグ
986
タグ/オペコード
待ち合わせステーション
1
実行ユニットおよび結果バスリクエスト
a オペランド
b オペランド
結果リクエスト
結果信号
945
b 転送
a 転送
955
950

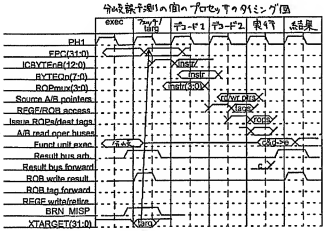
【図15】


836
ｲﾝｽﾄﾗｸｼｮﾝ 機能ユニット
885
命令ﾃﾞｺｰﾄﾞ
待ち合わせステーション
オペコード
オペランドバス
4
ﾃﾞｽﾃｨﾈｰｼｮﾝ Pred
ﾃﾞｽﾃｨﾈｰｼｮﾝ PC
835R
995
分岐予測比較
分岐 誤予測
正しいﾈｸｽﾄ PC
結果 が戻りおよび PC、結果バスへ
seq. PC
+1
990
分岐予測発生ﾋｽﾄﾘｰ
XTARGET
32
ICACHE 810 へ/から
980
結果バス
4

【図１６】



【図１７】



フロントページの続き

(72)発明者　デイビッド・ビィ・ウィット
アメリカ合衆国、78759　テキサス州、オースティン、パスファインダー・ドライブ、6318

(72)発明者　ウィリアム・エム・ジョンソン
アメリカ合衆国、78746　テキサス州、オースティン、クリスティ・ドライブ、102